Abbildung relationaler Daten auf die Ontologie des CIDOC CRM

Sven Peter

Masterarbeit

im Studiengang Angewandte Informatik

Fakultät Wirtschaftsinformatik und Angewandte Informatik
Otto-Friedrich-Universität Bamberg

Wissenschaftliche Betreuung: Prof. Dr. Christoph Schlieder
Softwaretechnischer Ansprechpartner: Peter Wullinger

Lehrstuhl für Angewandte Informatik
in den Kultur-, Geschichts- und Geowissenschaften

2015

URN: urn:nbn:de:bsz:16-artdok-34540

URL: http://archiv.ub.uni-heidelberg.de/artdok/volltexte/2015/3454

DANKSAGUNG

Ich möchte mich an dieser Stelle ganz besonders bei Herrn Prof. Dr. Christoph Schlieder bedanken. Er hat mir während meines Studiums mit Enthusiasmus die Fragestellungen und Möglichkeiten einer Informatik in den Kulturwissenschaften näher gebracht. Für die hervorragende Betreuung und das Interesse an meiner Arbeit gilt ihm ein großer Dank.

Ebenfalls ganz besonders möchte ich mich bei Herrn Moritz Schepp, dem Entwickler der ConedaKor-Datenbank, der mich während meines Praktikums am Deutschen Forum für Kunstgeschichte in Paris und darüber hinaus an seinem technischen Know-How teilhaben ließ und mich mit vielen wertvollen Ratschlägen bei meiner Arbeit unterstützt hat, bedanken.

Auch Herrn Thorsten Wübbena, dem Leiter der Abteilung „Digital Humanities“ am Deutschen Forum für Kunstgeschichte, sei für seine freundliche Aufnahme und den Einblicken, die er mir in die Fragestellungen einer digitalen Kunstgeschichte gewährte, gedankt.

Ebenso möchte ich mich bei Herrn Peter Wullinger bedanken, der mich mit wertvollen Hinweisen fachlicher wie auch organisatorischer Natur während meiner Masterarbeit begleitet hat.

Nicht zuletzt möchte ich mich auch bei meiner Familie und meinem Lebenspartner bedanken, die mich während meines Studiums unterstützt haben. Ohne sie wäre vieles nicht möglich gewesen.

Inhaltsverzeichnis

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

1 Einleitung

Auch in der geisteswissenschaftlichen Forschung und Lehre ist in den letzten Jahren unter dem Schlagwort „Digital Humanities“ vermehrt das Bewusstsein für den Nutzen digitaler Technologien erwacht.

Unter den sich ausdifferenzierenden digitalen Anwendungsgebieten für die geisteswissenschaftlichen Fachwissenschaften, beschäftigt sich die digitale Bildwissenschaft bzw. Kunstgeschichte[1] mit den digitalen Techniken und Methoden in den Bildwissenschaften [Kohle 2013].

Schon seit längerem wurden zur Erschließung gemeinsamer Arbeits- und Methodenfelder zwischen den Geisteswissenschaften und der Informatik interdisziplinäre Zentren für die Digitalen Geisteswissenschaften geschaffen, so in Trier[2], Göttingen[3] oder Köln[4].

Das Fach Kunstgeschichte ist bisher allerdings nur wenig in die digitalen Geisteswissenschaften an den Universitäten integriert. Das liegt zum einen daran, dass die Zuständigkeit bei dem direkten Umgang mit den Objekten mit denen sich die kunstgeschichtliche Forschung befasst nicht bei den Universitäten, sondern bei Institutionen wie Museen, Archiven oder Denkmalämtern liegt. Zum anderen sind allerdings auch erst seit kurzem die Rechnerleistungen ausreichend, um überhaupt eine sinnvolle Arbeit mit großen Mengen an Bilddaten zu ermöglichen.

Zwar wird sich nun auch in der Kunstgeschichte seit einiger Zeit der Digitalisierung von Kunstwerken und Literatur bedient, ein systematischer Einsatz digitaler Methoden erfolgt bisher allerdings nicht [Hoppe und Schelbert 2013]. Zu den bereits etablierten digitalen Werkzeugen, die die kunsthistorische Arbeit unterstützen, zählen OPACs wie *Kubikat*[5], Objektkataloge und Bilddatenbanken von Museen und Bildarchiven, aber auch soziale Medien wie Wikis und wissenschaftliche Blogportale wie *hypotheses.org*[6].

---

[1] Zum Verhältnis zwischen Bildwissenschaft und Kunstgeschichte beispielsweise siehe: Bredekamp, H. (2012): Kunstgeschichte als historische Bildwissenschaft. Verfügbar unter https://www.youtube.com/watch?v=ZcMjl0b3NG4 [20.02.2015].
[2] http://kompetenzzentrum.uni-trier.de/de [08.05.2015].
[3] http://www.gcdh.de/en [08.05.2015].
[4] http://www.cceh.uni-koeln.de/eHum [08.05.015].
[5] http://www.kubikat.org [22.10.2014].
[6] http://hypotheses.org [22.10.2014].

Werden diese neuen Technologien derzeit von der Mehrzahl der Kunsthistoriker noch hauptsächlich nur für die rasche Verbreitung von Informationen an ein möglichst großes Publikum über das Internet, beispielsweise durch die Verwendung eines Blogs, genutzt, bergen sie doch das Potential zu einem Paradigmenwechsel in der kunstgeschichtlichen Forschung [Cuno 2012].

Insbesondere ist die Nutzung kollaborativer Konzepte, die durch den Einsatz digitaler Technologien möglich werden, gerade in der Kunstgeschichte noch wenig verbreitet. Noch immer werden in der Kunstgeschichte Forschungsergebnisse individuell von jedem Forscher als abgeschlossenes Werk, dessen Genese sich über einen langen Zeitraum erstreckt, veröffentlicht. In den Naturwissenschaften ist dies schon seit längerem anders, indem durch die digitalen Technologien unterstützt, kollaborativ und inkrementell in schnellem Rhythmus Forschungsergebnissen veröffentlicht werden. An diese Arbeitsweise könnte sich auch die Kunstgeschichte anlehnen, indem sie Forschungsdaten in Form kleinster semantischer Einheiten als Mikropublikationen veröffentlicht, die so schnell zugänglich gemacht werden und dann auch quantitativ ähnlich wie in der naturwissenschaftlichen Forschung verarbeitet werden können.

Zur Realisierung dieses Paradigmenwechsels kunsthistorischer Arbeitsmethoden ist allerdings die Integration dieser Forschungsdaten nötig. Die Bemühungen zur Integration kunsthistorischer Datenbanken sind bisher noch durchaus dürftig. Lösungsansätze stehen in diesem Bereich in Form von Technologien wie Linked Data und Semantic Web sowie dem Einsatz von Normdaten, deren Erarbeitung im kunsthistorischen Bereich besonders vom Getty Research Institute[7] voran getrieben wird, zur Verfügung.

Auch am Deutschen Forum für Kunstgeschichte in Paris, einem Institut der Max-Weber-Stiftung, interessiert man sich mit der Schaffung einer Abteilung für die Digital Humanities 2014 intensiver für die Möglichkeiten digitaler Methoden und Werkzeuge für die kunstgeschichtliche Forschung, aber auch für neue Fragestellungen, die eine digitale Kunstgeschichte aufwirft.

Ein Projekt ist die Einführung eines neuen, graphbasierten Bilddatenbanksystems, ConedaKor[8], dessen Integration mit bereits an anderen kunstgeschichtlichen Instituten betriebenen Installationen desselben Datenbanksystems realisiert werden soll. Die Integration der Datenbanken soll den beteiligten Institutionen den verteilten Zugriff und die gemeinsame Nutzung der lokalen Datenbestände ermöglichen. Mit dem Ziele der Realisierung eines digitalen Werkzeuges zur Unterstützung der

[7] http://www.getty.edu/research [22.10.2014].
[8] http://coneda.net/ [05.01.2015].

Integration der ConedaKor-Datenbanken beschäftigt sich diese Arbeit mit der Problematik des Abbildens relationaler Daten auf die Ontologie des CIDOC CRM[9].

Nach dieser kurzen Einleitung soll in Kapitel 2 die Problemstellung der Arbeit präzisiert und die Datenmodelle der ConedaKor-Datenbank und des CIDOC CRM vorgestellt werden.

Kapitel 3 stellt das sogenannte Mapping als integrative Technologie zur Lösung semantischer Datenheterogenität, insbesondere in seiner Ausprägung als Mapping zwischen Schemata relationaler Datenbanken und Ontologien, vor und gibt einen Einblick in aktuelle Forschungsliteratur im Hinblick auf automatische Mapping-Ansätze.

Auf Grundlage der in Kapitel 3 vorgestellten Ansätze des Schema-Ontologie-Mappings diskutiert Kapitel 4 zunächst deren Einsetzbarkeit im Hinblick auf die konkrete Problemstellung der Arbeit. Im Anschluss daran wird der gewählte, softwaretechnisch zu realisierende Lösungsansatz dargelegt.

Die softwaretechnische Umsetzung des Mapping-Werkzeuges wird nach Funktionen strukturiert in Kapitel 5 beschrieben und die verwendeten Werkzeuge diskutiert.

Kapitel 6 stellt eine Evaluierungsmethode für das in Kapitel 5 beschriebene Mapping-Werkzeug vor, bevor die auf der Methode basierenden Evaluierungsergebnisse dargestellt und diskutiert werden.

Ein Fazit der Arbeit und einen Ausblick soll schließlich in Kapitel 7 gegeben werden.

2 Problemstellung

Die Realisierung der Interoperabilität der verschiedenen Installationen der ConedaKor-Datenbank erfordert sowohl die technische Integration der Systeme als auch eine semantische Integration der Daten.

Auf technischer Seite wird eine Peer-to-Peer-Lösung angestrebt, wobei diese Lösung gegenüber einer zentralisierten Lösung - beispielsweise in Form einer Client-Server-Architektur - den Vorteil hat, dass auf diese Weise Fragen der Zuständigkeiten zwischen den die verschiedenen zu vernetzenden Datenbanken bereitstellenden Institutionen weitestgehend vermieden werden können.

[9] http://www.cidoc-crm.org/ [14.05.2015].

Ebenso unerlässlich wie die technische Integration der Systeme ist auch die Integration der Daten. Über die syntaktische und strukturelle Integration hinaus stellt sich hier insbesondere die Frage der semantischen Integration, die durch eine unterschiedliche Konzeptualisierung der Daten in den unterschiedlichen, lokalen Datenbanken notwendig wird.

Im Rahmen dieser Arbeit soll zur Überwindung der semantischen Heterogenität der lokalen Datenbestände und zur Realisierung deren angestrebter Interoperabilität in der vernetzten Architektur der ConedaKor-Installationen die semantische Integration der Daten versucht werden. Im Zentrum dieses Vorhabens steht das Mapping - d.h. das Abbilden zwischen Elementen unterschiedlicher Datenmodelle - der in relationalen Datenbanken lokal vorgehaltenen Datengraphen von ConedaKor auf eine interoperable, globale Ontologie des kulturellen Erbes. Als prominente Instanz einer solchen Ontologie wurde das vom *Comité international pour la documentation* (CIDOC) des *Internatiol Council of Museum* (ICOM) entwickelte *CIDOC Conceptual Reference Model* (CIDOC CRM) gewählt, das seit 2006 auch internationaler Standard (ISO 21127) ist.

2.1 ConedaKor: eine Datenbank für Kulturerbe-informationen

ConedaKor ist ein quelloffenes Datenbanksystem[10] für den webbasierten Einsatz, das für die Archivierung, Verwaltung und Recherche von Bild- und Metadaten auf einer gemeinsamen graphischen Oberfläche entwickelt worden ist.

Technisch auf eine traditionelle relationale Datenbank aufbauend, ist ConedKor auf konzeptueller Ebene als Graphatenbank realisiert.

Der Datengraph stellt sich dabei als ein Netzwerk aus virtuellen Entitäten und deren Verbindungen zueinander dar. Die Entitäten lassen sich flexibel verbinden, wodurch es möglich wird komplexes und sich in seiner strukturellen Konzeptualisierung ständig weiter entwickelndes Wissen als dynamisches Netzwerk abzubilden. Ebenso ermöglicht die freie Anpassbarkeit die Kompatibilität mit anderen Datenmodellen, insbesondere dem relationalen, und erlaubt die einfache Migration bestehender Datensätze.

[10] https://github.com/coneda/kor [20.02.2015].

ConedaKor ist daher gerade für die Verwaltung von Informationen des kulturellen Erbes, die von hoher Komplexität und geringer Strukturiertheit geprägt sind, geeignet. So wird die Datenbank auch hauptsächlich von Institutionen, die Kulturerbeinformationen verwalten, wie den kunstgeschichtlichen Instituten der Goethe-Universität in Frankfurt/Main [11], der Universität des Saarlandes in Saarbrücken oder als Datenbank für kulturwissenschaftliche Forschungsprojekte wie dem Online-Projekt „Sandrart.net“, eingesetzt.[12]

Eine kurze durch Screenshots illustrierte Einführung in die Funktionalität von ConedaKor soll im Folgenden die im Rahmen dieser Arbeit wichtigen Konzepte und Begriffe von ConedaKor vorstellen. Hier wie auch im Folgenden bildet die am kunstgeschichtlichen Institut der Goethe-Universität in Frankfurt angelegte ConedaKor-Datenbank die Grundlage der in Bezug auf das Datenbanksystem gemachten Betrachtungen im Rahmen dieser Arbeit.

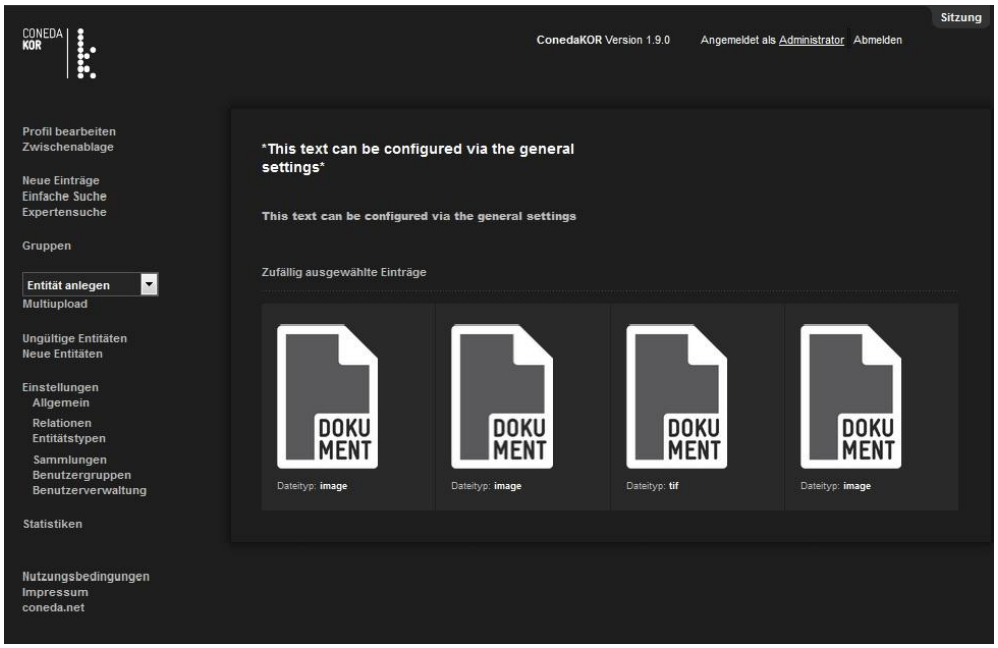

ABBILDUNG 1. CONEDAKOR-STARTBILDSCHIRM

In ConedaKor werden Entitäten und Verknüpfungen durch Entitätstypen (Abbildung 2) bzw. Relationen (Abbildung 3) typisiert. Es können neue Typen angelegt, und bereits bestehende Typen frei verändert oder auch gelöscht werden.

[11] https://kor.uni-frankfurt.de/ [20.02.2015].
[12] http://www.sandrart.net/de/ [20.02.2015].

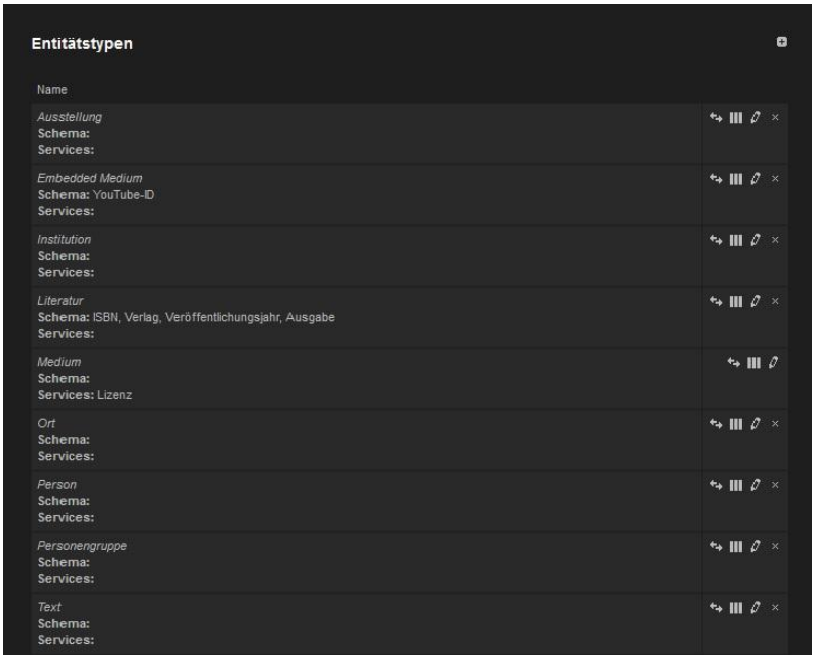

ABBILDUNG 2. LISTENDARSTELLUNG VON ENTITÄTSTYPEN

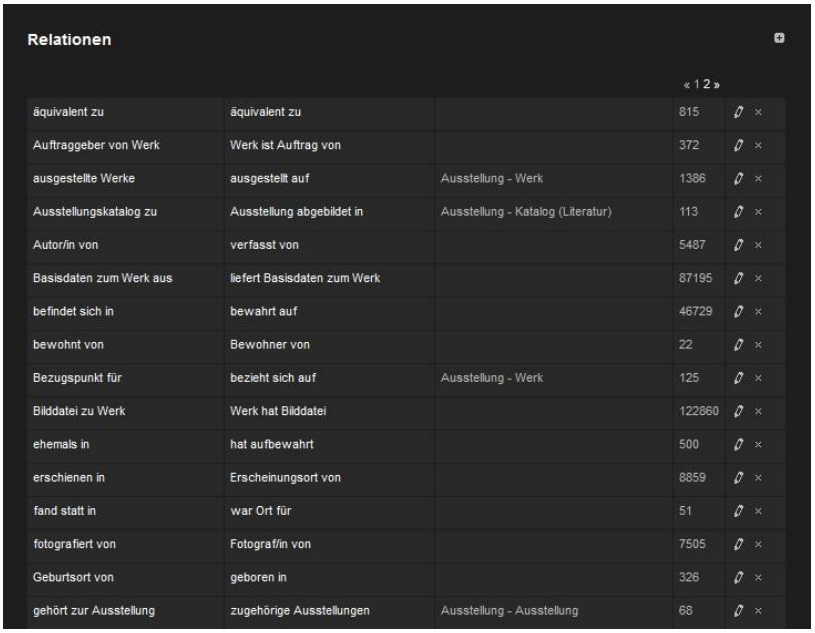

ABBILDUNG 3. LISTENDARSTELLUNG VON RELATIONEN

Relationen in ConedaKor sind ausschließlich binär, die von ihnen typisierten Verknüpfungen zwischen Entitäten bidirektional, was bedeutet, dass für eine angelegte Verknüpfung zwischen zwei Entitäten diese Verknüpfung von der einen

Entität zur anderen, als auch in umgekehrter Richtung navigierbar ist. Relationen sind daher durch einen ersten Namen, der die Semantik der Relation in Hinrichtung, und einen zweiten Namen, der die Semantik der Relation in Rückrichtung ausdrückt, definiert (Abbildung 4). Darüber hinaus sind Hin- und Rückrichtung durch jeweils erlaubte Entitätstypen - im Folgenden Start- und Zielentitätstypen genannt -, deren Entitäten über Verbindungen der Relation verbunden werden können, weiter spezifiziert. Zusätzlich zur Benennung kann die Semantik der Relation durch eine Beschreibung weiter präzisiert werden.

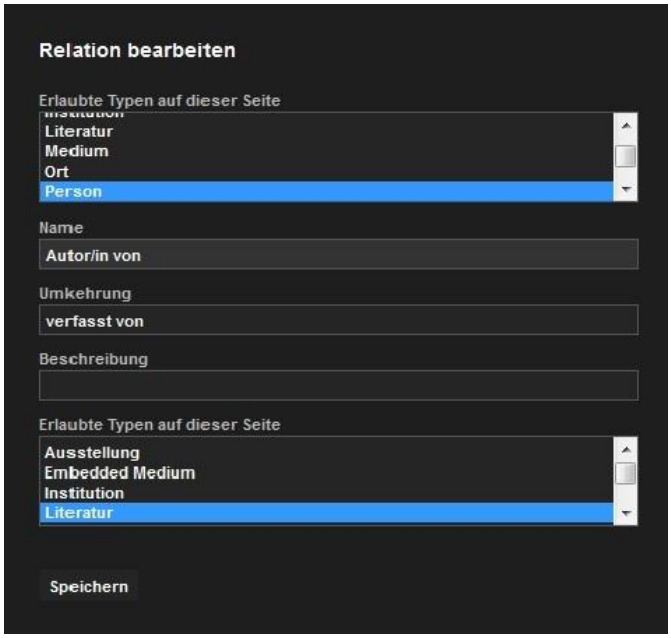

ABBILDUNG 4. MASKE ZUR BEARBEITUNG EINER RELATION

Die freie Anpassbarkeit bedeutet wie bereits erwähnt nicht nur die Möglichkeit neue Entitätstypen oder Relationen anlegen zu können, sondern es können diese auch gelöscht oder bearbeitet werden. Modifikationen der Typen wirken sich dabei allerdings nicht auf die bereits angelegten Entitäten und Verknüpfungen aus. So ist es in der aktuellen Version von ConedaKor[13] beispielsweise möglich Start- und Zielentitätstypen einer Relation nachträglich zu ändern. Die bereits existierenden Verknüpfungen dieser Relation zwischen Entitäten bleiben unverändert bestehen, auch wenn deren Entitätstypen nicht mehr den aktuell definierten Start- und Zielentitätstypen der Relation entsprechen.

[13] Version 1.8.3.

Entitätstypen werden durch Namen bezeichnet, die die Semantik des jeweiligen Entitätstyps ausdrücken (Abbildung 5). Durch eine Beschreibung kann deren Semantik weiter präzisiert werden. Über die Angabe von Web Services werden die Entitäten des jeweiligen Typs mit anderen Diensten, die die von der jeweiligen virtuellen Entität in ConedaKor repräsentierten realweltliche Entität ebenfalls virtuell repräsentieren, verlinkt. Jeder Entitätstyp hat des Weiteren drei vordefinierte Attribute oder Felder für Datierung, Namen sowie eindeutigen Namen der Entitäten des Entitätstyps. Die Bedeutung dieser Felder können durch die Angabe einer Bezeichnung für jeden Entitätstyp präzisiert werden.

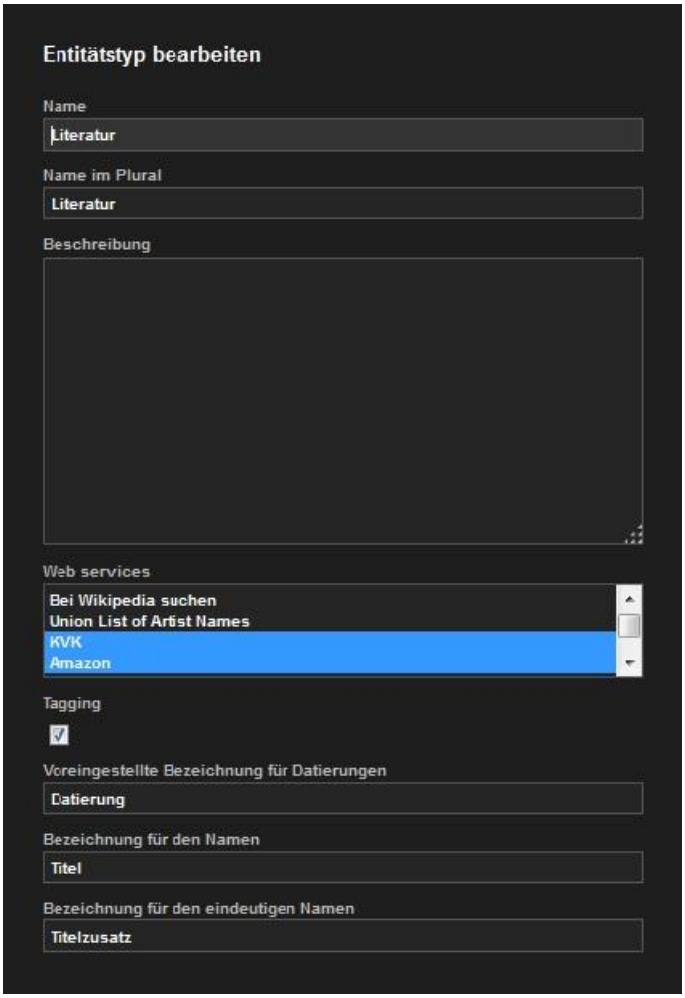

ABBILDUNG 5. MASKE ZUR BEARBEITUNG EINES ENTITÄTSTYPS

Zusätzlich zu den festen Attributen oder Feldern für jeden Entitätstyp können weitere, spezifische Felder für jeden Entitätstyp definiert werden. Im Falle des Entitätstyps *Literatur* könnten dies beispielsweise Felder für die ISBN, den Verleger, das Erscheinungsdatum oder die Edition sein (Abbildung 6).

ABBILDUNG 6. FELDER DES ENTITÄTSTYPS *LITERATUR*

Jede Entität besitzt somit einen Entitätstyp, der Felder seiner Entitäten definiert. Zusätzliche Felder können aber auch individuell für jede Entität definiert werden. Im unten stehenden Beispiel des Bamberger Altars von Veit Stoß (Abbildung 7) handelt es sich um eine Entität des Typs *Werk*. Die Entität hat die Felder *Synonyme*, *Datierung*, *Material und Technik* und ist über die Verknüpfungen *Basisdaten zum Werk*, *ist Teil von*, *steht in Verbindung zu* und *wurde geschaffen von* mit weiteren Entitäten, wie der Person *Stoß, Veit*, verbunden (Abbildung 8).

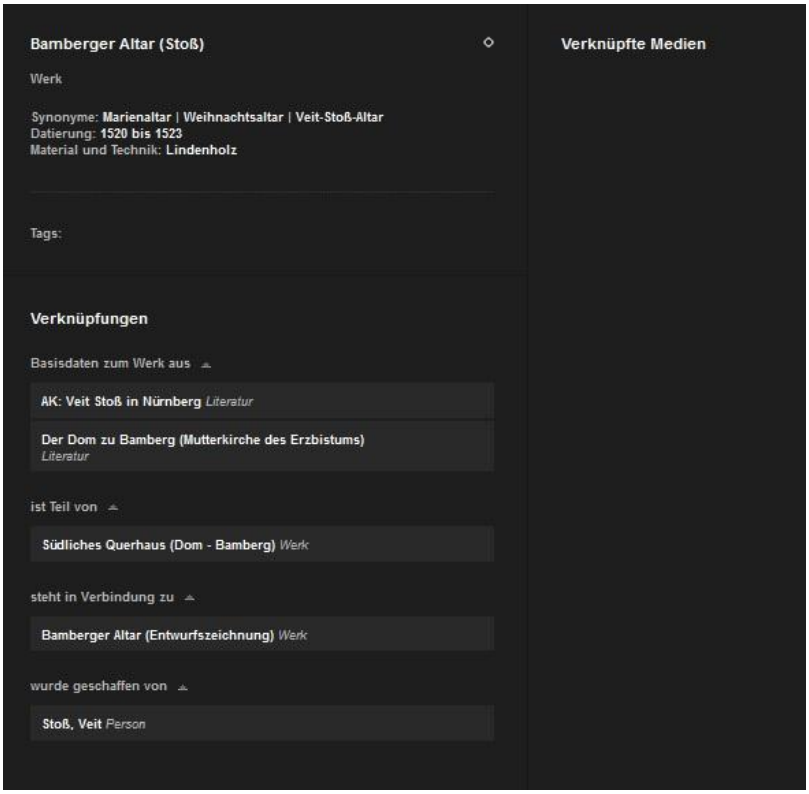

ABBILDUNG 7.DARSTELLUNG DER ENTITÄT *BAMBERGER ALTAR (STOß)*

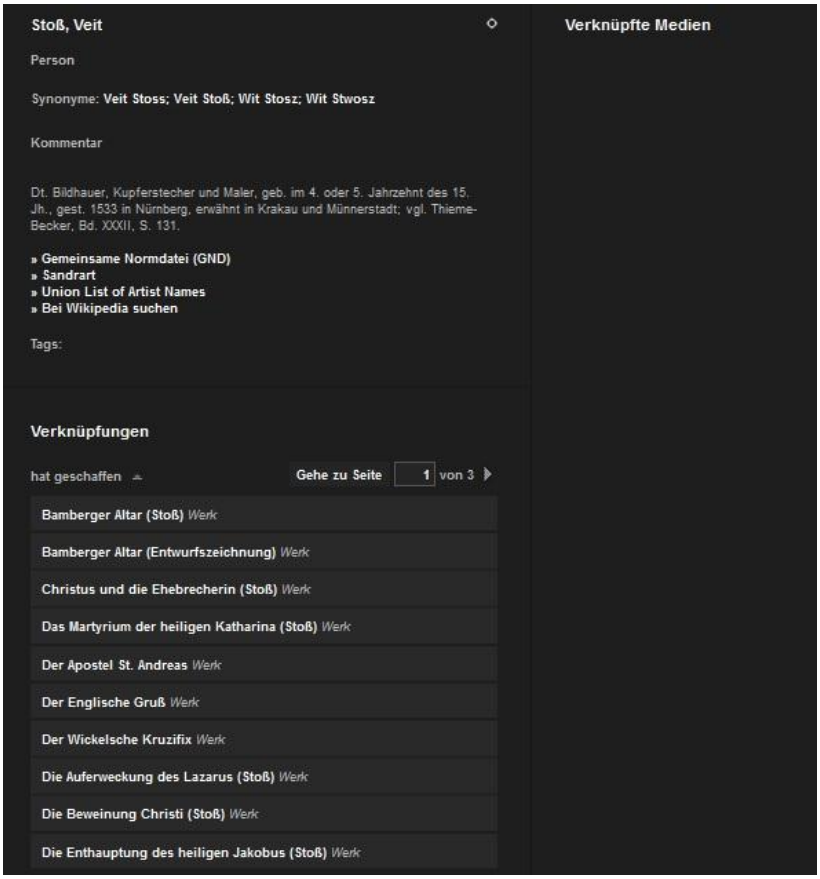

ABBILDUNG 8. DARSTELLUNG DER ENTITÄT *STOß, VEIT*

2.1.1 ConedaKor: eine graphbasierte Datenbank

Wie bereits angesprochen ist die die ConedaKor-Datenbank zwar technisch als relationale Datenbank realisiert, auf konzeptueller Ebene handelt es sich allerdings um eine Graphdatenbank.

Graphdatenbanken eignen sich besonders für vernetzte, semistrukturierte und dynamische Datensätze und bieten damit eine Alternative zu anderen verbreiteten Datenbankarten, wie den relationalen oder objektorientierten Datenbanken.

Die Tatsache, dass sich viele Anwendungsgebiete auf das den Graphdatenbanken zugrunde liegende flexible Graphmodell [Angeles und Gutierrez, 2008] abbilden lassen, hat gerade bei neuartigen Anwendungen mit hochgradig vernetzten Daten, wie sie beispielsweise im Semantic Web üblich sind, die Verwendung von Graphdatenbanken beflügelt. Denn dort ist die Flexibilität der Graphdatenbank, die semistrukturierte Datensätze aus unterschiedlichen verteilten Datenquellen und Datenschemata unterstützt, besonders gefordert. Gerade im Vergleich zu

relationalen Datenbanken, die mit ihren statischen Datenschemata nur schlecht für semistrukturierte, evoluierende Datensätze aus verschiedenen Datenquellen geeignet sind, zeichnen sich Graphdatenbanken auch durch eine weit höhere Effizienz bei der Traversierung gerade großer Datenmengen aus. So verursachen bei traditionellen relationalen Datenbanken die zur Traversierung nötigen Bildungen des kartesischen Produkts von Relationen gerade bei großen vernetzten Datenmengen einen erheblichen Rechenaufwand.

Ein Graph [Heckel 2006; Rodriguez und Neubauer 2010] besteht im Wesentlichen aus Knoten und Kanten, die diese Knoten miteinander verbinden. Er lässt sich somit formal als ein Tupel $G = (V, E)$ darstellen, wobei V die Menge der Knoten und E die Menge der Kanten zwischen diesen Knoten, also $E \subseteq V \times V$, ist. Es gibt zwei Arten von Kanten, gerichtete und ungerichtete. Bei einer gerichteten Kante ist die Reihenfolge der durch die Kante verbundenen Knoten von Bedeutung, d.h. die Kante definiert eine Verbindung in eine Richtung, von einem Start- zu einem Zielknoten. Bei einer ungerichteten Kante ist die Verbindung ohne zusätzliche Einschränkung einer Richtung zwischen zwei Knoten definiert. Eine ungerichtete Kante kann somit auch als zwei gerichtete Kanten, eine vom Start- zum Zielknoten und eine vom Ziel- zum Startknoten, interpretiert werden, d.h. $(v1, v2) \in E \Rightarrow (v2, v1) \in E$.

Eine Erweiterung dieses Modells eines Graphen ist der gewichtete Graph. Hier wird zusätzlich eine Funktion $w: E \rightarrow \mathbb{R}$ definiert, die jeder Kante einen numerischen Wert zuordnet, der oft die Kosten der Navigation, d.h. des Abschreitens des Graphen von einem Knoten zum andern, über diese Kante ausdrücken soll.

Das Modell des gewichteten Graphen kann weiter verallgemeinert werden, indem keine Funktion, die nur den Kanten des Graphen reelle Werte zuordnet, sondern eine Funktion, die Kanten und Knoten beliebige Informationen zuordnet, definiert wird. Die zugeordneten Informationen können weiter durch Schlüssel-Wert-Paare strukturiert werden, bei denen jedem eindeutigen Schlüssel ein Wert zugeordnet wird. Eine solche Funktion kann durch $\mu: (V \cup E) \times S \rightarrow (T \cup \{\emptyset\})$, mit S ist die Menge aller Schlüssel und T ist die Menge aller Werte, formalisiert werden. Die den Knoten und Kanten so zugeordneten Schlüssel-Wert-Paare nennt man Attribute, ein solcher Graph attribuierter Graph.

Ein Spezialfall der oben beschriebenen Attribuierung weist der bezeichnete Graph auf. Hier werden Funktionen $\lambda V: V \rightarrow L,\ \lambda E: E \rightarrow V$ definiert, die jeder Kante und jedem Knoten eine bestimmte Bezeichnung $lx \in L$ zuweisen. Auf diese Weise lässt sich beispielsweise die Typisierung der Knoten und Kanten realisieren, indem Knoten und Kanten Bezeichnungen zugewiesen werden, die für den Typ des Knotens oder der Kante stehen.

Zusammenfassend lässt sich ein bezeichneter, attribuierter Graph durch folgende mathematische Struktur formalisieren:

- $G = (V, E, S, T, \lambda V, \lambda E, \mu)$ ist ein bezeichneter, attribuierter Graph
- V ist die Menge aller Knoten des Graphen
- $E \subseteq V \times V$ ist die Menge aller Kanten des Graphen
- S ist die Menge aller Attributschlüssel
- T ist die Menge aller Attributwerte
- $\lambda V: V \to L$ ist eine Knotenbezeichnung
- $\lambda E: E \to V$ ist eine Kantenbezeichnung
- $\mu: (V \cup E) \times S \to (T \cup \{\emptyset\})$ ist eine Schlüssel-Wert-Zuordnung

Ein so definierter Graph ist zunächst keinen weiteren Einschränkungen unterworfen und so können sowohl beliebige Knoten über beliebige Kanten miteinander verknüpft, als auch Knoten und Kanten beliebige Attribute und Bezeichner frei zugeordnet werden. Man nennt diese Art Graph schemafrei. Oft ist es jedoch sinnvoll die strukturelle Freiheit eines Graphen einzuschränken, da man gewisse Strukturen ausschließen möchte. Zu diesem Zweck wird ein Schema für den Graphen definiert, das beispielsweise die möglichen Beziehungen zwischen Kanten- und Knotentypen oder die zulässigen Attribute der Knoten und Kanten definiert. Ein solches Schema kann selbst ebenfalls als Graph modelliert werden.

Im Folgenden sollen diese generellen Überlegungen zum Graphmodell am Datenmodell der ConedaKor-Datenbank illustriert und letzteres damit weiter beschrieben und formalisiert werden.

Die Daten der ConedaKor-Datenbank können auf konzeptueller Ebene als bezeichneten, attribuierten Graph modelliert werden. In Abbildung 9 ist ein Ausschnitt aus dem Datengraph der Frankfurter ConedaKor-Datenbank dargestellt. Der Graph besteht aus Entitätsknoten (blau) und ungerichteten Verknüpfungskanten (gelb). Entitäten und Kanten sind bezeichnet und damit typisiert (rot für Entitätstypen, grün für Relationen). Auch besitzen Entitäten Attribute (grau), deren Schlüssel wie bereits oben angesprochen teilweise durch den Typ der jeweiligen Entität und teilweise frei für die individuelle Entität definiert sind.

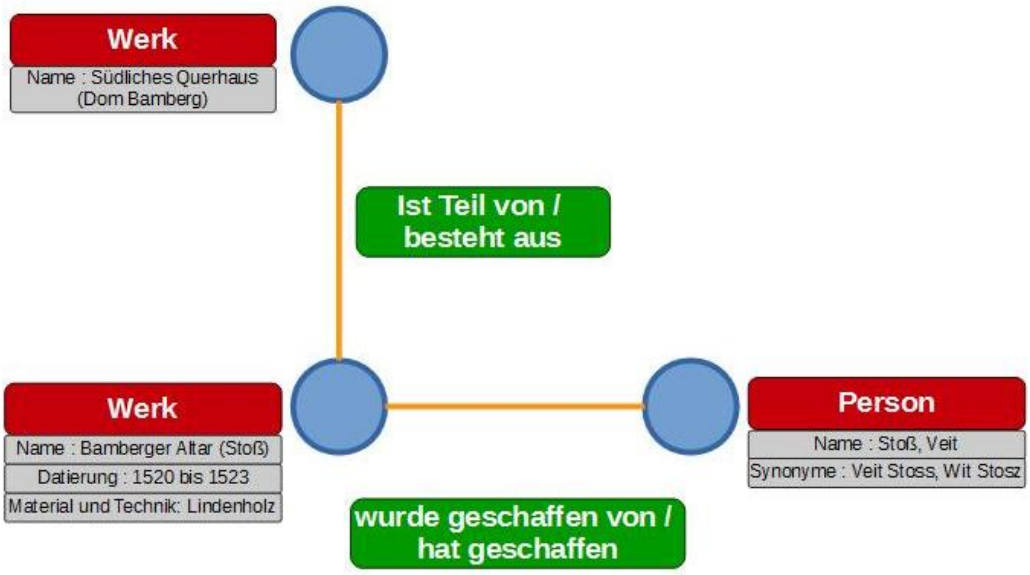

ABBILDUNG 9. AUSSCHNITT AUS DEM DATENGRAPH DER FRANKFURTER CONEDAKOR-DATENBANK

Aus dem oben dargestellten Instanzgraphen kann auf Grundlage der Bezeichnung der Knoten und Kanten, die informal deren Semantik beschreiben, ein Schemagraph abgeleitet werden, der die Entitätstypen und Relationen als Knoten bzw. Kanten enthält und die aus dem Instanzgraphen abgeleiteten Abhängigkeiten zwischen diesen abbildet (Abbildung 10).

In diesem Schemagraphen sind die im Schemagraphen freien Typen auf die Typen *Entitätstyp* und *Relation* für Knoten sowie den Typ *Von / zu Entitätsart* reduziert. Ebenso sind die möglichen Attribute eingeschränkt. Gerade die aus dem Instanzgraphen abgeleiteten Start- und Zielentitätstypen einer Relation sind interessante Informationen auf Schemaebene. Dieser Schemagraph bildet die von der Instanzebene auf die Typenebene abstrahierte Struktur des Datengraphen in der ConedaKor-Datenbank ab.

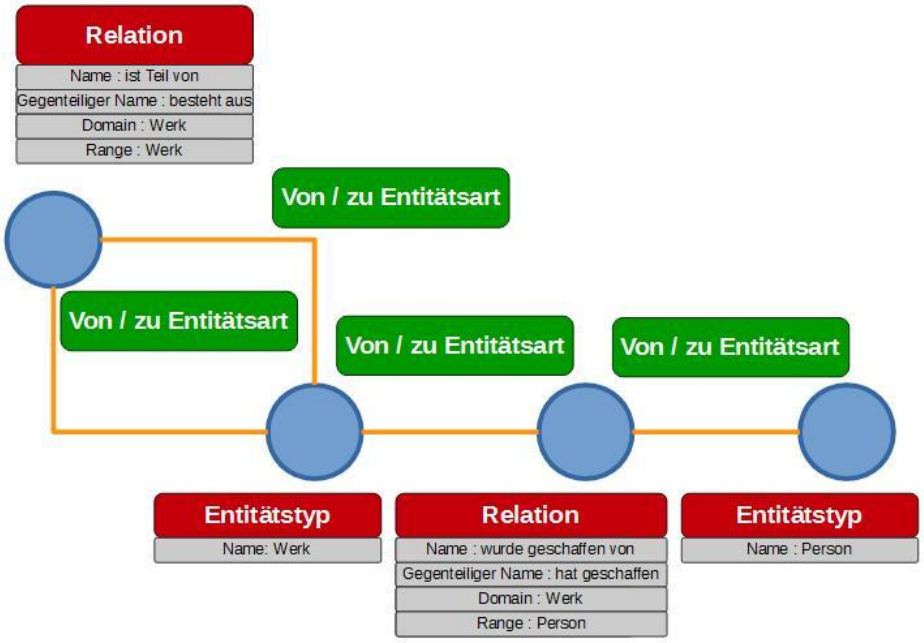

ABBILDUNG 10. AUS OBIGEM INSTANZGRAPHEN ABGELEITETER SCHEMAGRAPH

2.1.2 ConedaKor: Eine relationale Datenbank

Datenbanken werden in der Regel in drei Schritten entworfen. Am Anfang steht der konzeptuelle Entwurf der Datenbank, der auf einer Analyse der Anforderungen an die Daten und die auf diesen operierenden Funktionen beruht. Diese Entwurfsphase mündet in eine Spezifizierung in Form eines sogenannten konzeptuellen Schemas, das in Form besonderer Modelle wie beispielsweise dem Enitity-Relationship-Modell (ERM) [Chen 1976] ausgedrückt werden kann. Im Falle von ConedaKor steht auf konzeptueller Schemaebene das oben bereits dargestellte Graphmodell.

Zwischen dem konzeptuellen und dem physischen Entwurf der Datenbank, der letztlich das eigentliche Abbild der Datenbank im physischen Speicher der Maschine festlegt, steht der logische Entwurf. Durch ein passendes Repräsentationsmodell ausgedrückt, steht am Ende dieser Entwurfsphase das logische Schema der Datenbank.

Auch hier gibt es verschiedene Modelle, um die Datenbank auf logischer Ebene zu repräsentieren. Das wohl bekannteste darunter ist das Relationale Datenbankmodell [Codd 1970]. Dieses erlaubt die Modellierung der Daten durch Relationen und deren Attribute, die durch Tabellen mit zugehörigen Spalten dargestellt werden können. Jedes Attribut kann dabei nur Werte aus einer vordefinierten Wertemenge annehmen. Beziehungen werden in diesem Modell über spezielle Attribute, Schlüssel

genannt, realisiert, die es zum einen erlauben ein Tupel einer Relation eindeutig zu identifizieren (Primärschlüssel) und zum anderen eine Beziehung zu einem anderen Tupel herzustellen (Fremdschlüssel). Vorteil des logischen im Vergleich zum konzeptuellen Datenschema ist, dass es mit den Operatoren der relationalen Algebra verarbeitet werden kann, die somit eine standardmäßige Schnittstelle auf die Daten auf dieser Ebene spezifiziert.

Im relationalen Datenmodell werden die Daten als Relationen dargestellt. Eine Relation kann mathematisch definiert werden als $R \subseteq D1 \times D2 \times \ldots \times Dn$, wobei $D1, D2, \ldots,$ Dn Mengen von Werten sind. Ein Element $r = (d1, d2, \ldots, dn)\ \epsilon\ R$ mit $(di \in Di, i = 1, \ldots, n)$ heißt Tupel und beschreibt in der Regel eine Entität. Die Eigenschaften einer Relation, d.h. also einer gleichartigen Menge von Tupeln, werden durch ein Relationenschema beschrieben. Ein Relationenschema $R(A1, \ldots, An)$ spezifiziert eine Relation mit dem Namen R und den Attributen $A1, A2, \ldots, An$. Jedem dieser Attribute ist eine Wertebereich $dom(Ai)$ zugeordnet. Die zum Schema $R(A1, \ldots, An)$ gehörigen Relationen sind also alle vom Typ $R \subseteq dom(A1) \times dom(A2) \times \ldots \times dom(An)$. Zu jedem Zeitpunkt existiert in der Datenbank immer genau eine Relation R zum Relationenschema $R(A1, \ldots, An)$.

In Tabelle 1 ist das relationale Datenschema der im Rahmen dieser Arbeit verwendeten Frankfurter ConedaKor-Datenbank ausschnitthaft für die im Rahmen dieser Arbeit relevanten Daten dargestellt. Dazu gehören Entitätstypen (*kinds*), Relationen (*relations*), Entitäten (*entities*), Verknüpfungen (*relationships*), Entitätsdatierungen (*entity_datings*), sowie weitere Entitätsattribute, die hier jedoch nicht weiter betrachtet werden sollen:

Relation	**Attribute**
kinds	*id, name*
relations	*id, name, reverse_name*
entities	*id, name, distinct_name, kind_id*
relationships	*id, relation_id, from_id, to_id*
entity_datings	*id, entity_id, label, dating_string*

TABELLE 1. CONEDAKOR-RELATIONEN

Die Attribute *entities.kind_id*, *relationships.relation_id*, *relationships.from_id*, *relationships.to_id* und *entity_datings.entity_id* sind Fremdschlüssel und verweisen auf die Primärschlüssel der zugehörigen Relationen.

2.2 Das CIDOC CRM: eine Ontologie für Kulturerbe-informationen

Das Conceptual Reference Model (CRM) ist das Ergebnis der 1996 begonnenen Bemühungen des *Comité international pour la documentation* (CIDOC) des *International Council of Museums* (ICOM), eines weltweiten Zusammenschlusses von Museen und ähnlichen Einrichtungen, zur Erarbeitung einer die Informationen des kulturellen Erbes repräsentierenden formalen Ontologie. Die Ontologie ist als High-Level-Ontologie zur Informationsintegration von Kulturerebedaten entworfen worden [Doerr 2003; 2009] und wurde dabei zum großen Teil aus den bereits existierenden Datenstrukturen, die in Kulturerbeinstitutionen zur Informations-dokumentation verwendet werden abgeleitet. Seit 2006 ist das CIDOC CRM internationaler Standard (ISO 21127). Seit 2014 existiert es mit 87 Klassen (Classes) und 263 Eigenschaften (Properties) als Version 5.0.4 [Crofts 2011].

2.2.1 Zum Ontologiebegriff

Als philosophischen Disziplin in der Bedeutung der Lehre von Natur und Struktur der Realität verwendet, steht der Begriff Ontologie (von gr. ὤν, ὄντος „seiend“, Partizip Präsens des Verbs εἰμί „sein“, und gr. Λόγος „Lehre“) im informationswissenschaftlichen Kontext für ein Mittel zur formalen Modellierung der Struktur eines Systems von Entitäten als Konzepte und deren Relationen [Guarino, Oberle und Staab 2009]. Grundlage von Ontologien in diesem Sinne sind in der Regel Taxonomien, d.h. Generalisierungs- und Spezialisierungshierarchien dieser Konzepte.

Die Definition des Begriffs Ontologie im informationswissenschaftlichen Sinne wurde über die Jahre Schritt für Schritt präzisiert. Gruber definierte den Begriff Ontologie 1993 als „*explicit specification of a conceptualization*“ (eine explizite Spezifizierung einer Konzeptualisierung) [Gruber 1993]. Borst fügte 1997 mit seiner Definition einer Ontologie als „*formal specification of a shared conceptualization*“ (eine formale Spezifizierung einer gemeinsamen Konzeptualisierung) die Ideen hinzu, dass die Konzeptualisierung auf einem möglichst breiten Konsens und nicht nur auf einer individuellen Auffassung beruhen soll und dass die Spezifizierung formal, d.h.

maschinell les- und verarbeitbar sein soll [Borst 1997]. Studer, Benjamins und Fensel führten schließlich 1998 beide Definitionen zusammen in der Aussage „*An ontology is a formal, explicit specification of a shared conceptualization*“ (Eine Ontologie ist eine formale, explizite Spezifizierung einer gemeinsamen Konzeptualisierung) [Studer, Benjamins und Fensel 1998].

Insbesondere im Rahmen des Semantic Web werden Ontologien als Grundlage interoperabler Wissensrepräsentation genutzt wird.

Das Semantic Web geht dabei auf eine Idee Tim Berners-Lee [Berners-Lee 1998; 2001], dem Erfinder des World Wide Webs und Direktor des W3C, zurück. In der immer größer werdenden Menge von Information im Internet die Gefahr eines für den Menschen nicht mehr beherrschbaren Informationsüberflusses. So kündigte Google bereits im Juli 2008 an, dass ihre Webcrawler bereits eine Trillion Seiten gefunden hätten (während es 1998 “nur” 26 Millionen waren)[14]. Und dabei sind die dynamisch generierten Inhalte des sogenannten Deep Webs, d.h. des nicht von Web-Suchmaschinen erfassbaren Teil des World Wide Webs, die immer bedeutender werden nicht einmal mit eingerechnet. Bei dieser vom Menschen kognitiv nicht mehr beherrschbaren Informationsfülle können automatische Methoden Abhilfe schaffen. Die Voraussetzung der automatischen Verarbeitung der Informationen im Internet ist der Übergang des auf unstrukturierten, für die Verarbeitung durch den Menschen gedachten Webs der Dokumente zu einem formal strukturierten und damit potentiell automatisch verarbeitbaren Webs der Daten.

Eine adäquate Wissensrepräsentation bietet dabei nicht nur die Möglichkeit der maschinellen Verarbeitung dieses Wissens, sondern erlaubt dabei auch die Integration bisher unerschlossener Wissensbasen. So können Daten, die beispielsweise in digitalen Archiven und Bibliotheken, Datenbanken von Behörden, usw. liegen und die im derzeitigen Web der Dokumente dem Zugang über Websuchmaschinen größtenteils entzogen sind, in einer datenorientierten Form der Wissensrepräsentation in Form eines Webs der Daten in eine globale Wissensbasis integriert werden.

Eine solche Form der Wissensrepräsentation, die also den integrierten Zugriff auf heterogene, autonome und verteilte Informationsquellen und deren automatische Verarbeitbarkeit erlaubt, kann durch Ontologien realisiert werden.

[14] http://googleblog.blogspot.fr/2008/07/we-knew-web-was-big.html [20.11.2014].

2.2.2 Ontologierepräsentation

Zur Repräsentation von Ontologien gibt es verschiedene Technologien. Neben Topic Maps ist eine besonders häufige Art der Ontologierepräsentation die Beschreibung der Ontologie durch RDF, RDFS (RDF-Schema) und OWL (Web Ontology Language)[15].

Das Resource Description Framework (RDF) stellt ein graphbasiertes Modell dar. Ein RDF-Graph besteht aus einer Menge von RDF-Tripeln, so genannten Aussagen, die einen gerichteten markierten Graphen bilden. Eine Aussage besteht immer aus drei Teilen, dem Subjekt über das die Aussage gemacht wird, dem Prädikat und dem zugehörigen Objekt. Das Subjekt ist immer eine Ressource, d.h. irgendetwas, ob Konkretum oder Abstraktum, über das eine Aussage getroffen werden kann und das mittels eines URI (Uniform Ressource Identifiers) eindeutig identifiziert wird.

Der RDF-Graph kann auf verschiedene Arten repräsentiert werden. Eine für den Menschen leicht verständliche Repräsentationsart ist das Graphdiagramm. Für die Serialisierung und maschinelle Verarbeitung ist die XML basierte Repräsentation des RDF-Graphen weit verbreitet.

RDFS definiert ein Vokabular mit festgelegter Bedeutung, welches in RDF-Datenmodellen Verwendung findet und deren Struktur im Sinne eines Schemas festlegt. Die von RDFS bereitgestellten Schemaprimitive erlauben beispielsweise die Definition von Klassenhierarchien oder die Definition von Start- und Zielklassen von Eigenschaften. RDFS ist somit selbst bereits eine primitive Ontologiesprache, die ebenfalls als RDF-Graph repräsentiert werden kann.

OWL schließlich ist eine noch weitaus mächtigere Ontologiesprache, die in drei Varianten zur Verfügung steht. Diese sind nach zunehmender Ausdruckskraft geordnet die folgenden: OWL Lite, OWL DL und OWL Full. OWL setzt sich dabei aus Axiomen, die Klassen und Eigenschaften näher beschreiben, und Fakten, die Individuen näher beschreiben, zusammen. Die in OWL zur Verfügung stehenden Konstrukte basieren auf der formalen Logik und so können aus den definierten Abhängigkeiten der explizit kodierten Informationen durch Inferenz weitere implizite Informationen abgeleitet werden. Auch OWL ist im RDF-Graph repräsentierbar.

[15] Eine praktische Einführung ist Allemang, D., & Hendler, J. (2011). *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier.

2.2.3 Ziele des CRM

Das CIDOC CRM definiert die Semantik von Kulturerbeinformationen und bietet ein objektorientiertes Modell der intellektuellen Strukturen kultureller Dokumentation in Begriffen der Logik.

Das CIDOC CRM legt dabei die Beziehungen der Modellelemente zueinander, nicht allerdings deren Terminologie festlegt. Ebenso versteht sich das CRM generisch und ist daher für keine spezifische Implementation optimiert. Ebenso kann das CRM erweitert für die jeweiligen Bedürfnisse erweitert werden.

Dadurch lässt sich durch das CIDOC CRM die Informationsinteroperabilität auf semantischer Ebene erzielen und dient somit als Basis für den Austausch von Kulturerbeinformationen und die Transformation von lokalen, heterogenen Informationsquellen in eine integrierte, globale Wissensressource.

Das CRM hat verschiedene Anwendungen. Es unterstützt als Vorbild die konzeptuelle Modellierung neuer Informationsquellen. Es dient als Referenz zur Kommunikation zwischen Domänenexperten und Softwareentwicklern. Als formale Sprache unterstützt es die Implementation automatischer Datentransformationsalgorithmen, die lokale in globale Datenstrukturen überführen. Es unterstützt des Weiteren die Formulierung assoziativer Anfragen an integrierte Informationsressourcen, indem es ein globales Modell bereitstellt.

Wie bereits erwähnt repräsentiert das CRM das gesamte kuratierte Wissen von Museen. Diesem intendierten Geltungsbereich, der somit alle Informationen, die zum Austausch und zur Integration der heterogenen wissenschaftlichen Dokumentationsstrukturen der Museumskollektionen erforderlich sind, beinhaltet, steht der praktische Geltungsbereich gegenüber.

Der praktische Geltungsbereich wird durch die jeweiligen aktuellen Dokumentationsstandards der Museen ausgedrückt, die in die Entwicklung und Validierung des CIDOC CRM eingeflossen sind. Daten, die diesen Dokumentationsstandards gehorchen, können in CRM-kompatibler Form ausgedrückt werden.

2.2.4 Modell und Struktur des CRM

Das CIDOC CRM hat einen objektorientierten Aufbau. Einige essentielle Konzepte des objektorientierten Paradigmas, das besonders prominent von vielen erfolgreichen Programmiersprachen insbesondere seit den neunziger Jahren umgesetzt wird, sollen im Folgenden erläutert werden.

Besonders zentrale Konzepte der Objektorientierung sind Klassifikation und Identität. Das objektorientierte Paradigma unterscheidet zwischen Objekten, die auch wenn sie ihren Zustand ändern ihre Identität bewahren und Klassen, die die Eigenschaften von Gruppen von Objekten beschreiben und somit eine Abstraktion der konkreten Objekte darstellen. Man spricht daher bei einem Objekt auch von einer Instanz einer Klasse.

Zwischen den Klassen bestehen im objektorientierten Modell Spezialisierungs- und Generalisierungsbeziehungen, d.h. es existiert eine Hierarchie von Klassen übergeordneten Super- bzw. untergeordneten Subklassen (auch Ober- bzw. Unterklassen genannt). Über diese Beziehungen vererben Superklassen ihre Eigenschaften an die sie spezialisierenden Subklassen. Eine Konsequenz dieser Hierarchisierung von Klassen ist auf Instanzebene die sogenannte Polymorphie von Objekten, d.h. die Zugehörigkeit von Instanzen von Klassen zur Klasse selbst sowie zu allen der Klasse übergeordneten, abstrakteren Superklassen.

Zwei wichtige Relationen auf Instanzebene im objektorientierten Modell sind die sogenannte Ist-Ein-Relation und die sogenannte Hat-Ein-Relation. Die Ist-Ein-Relation drückt dabei die Zugehörigkeit eines Objekts zu einer Klasse, d.h. seines Typs, oder auch einer Klasse zu einer anderen Klasse, d.h. in diesem Fall der Superklasse der Klasse, aus. Die Hat-Ein-Relation stellt dazu im Gegensatz eine Beziehung zwischen einem Objekt und einem weiteren Objekt dar.

Als ein solches objektorientiertes Modell ist, wie bereits erwähnt, das CIDOC CRM realisiert. Die zentralen Elemente des CRM sind Klassen und Eigenschaften (Properties).

Das CRM ist hierarchisch aus Klassen aufgebaut. Die verschiedenen Klassen sind durch einen Code mit Präfix „E“ und darauffolgender ganzer Zahl benannt (der Buchstabe E zur Bezeichnung der Klassen beruht auf einer früheren Modellierung des CIDOC CRM als Entity-Relationship-Modell). Weiter hat eine Klasse auch eine natürlichsprachliche Bezeichnung in Form einer Nominalgruppe, die allerdings nicht wie der Code zur Identifizierung der Klasse genutzt wird, um die Identifizierung unabhängig von einer spezifischen Sprache zu halten.

Die Klassen sind nicht formal sondern natürlichsprachlich durch eine sogenannte *Scope Note*, d.h. eine Intension, die die Semantik des Konzeptes der Klasse ausdrückt, sowie den Properties der Klasse definiert. Die definierten Properties einer Klasse werden an ihre Subklassen in der Klassenhierarchie vererbt. Instanzen der Klassen, d.h. deren Extensionen, auch Entitäten genannt, können über die Properties der Klassen, deren Instanzen sie sind, mit weiteren Entitäten verbunden werden, wodurch die Semantik der Entitäten zunehmend reicher wird.

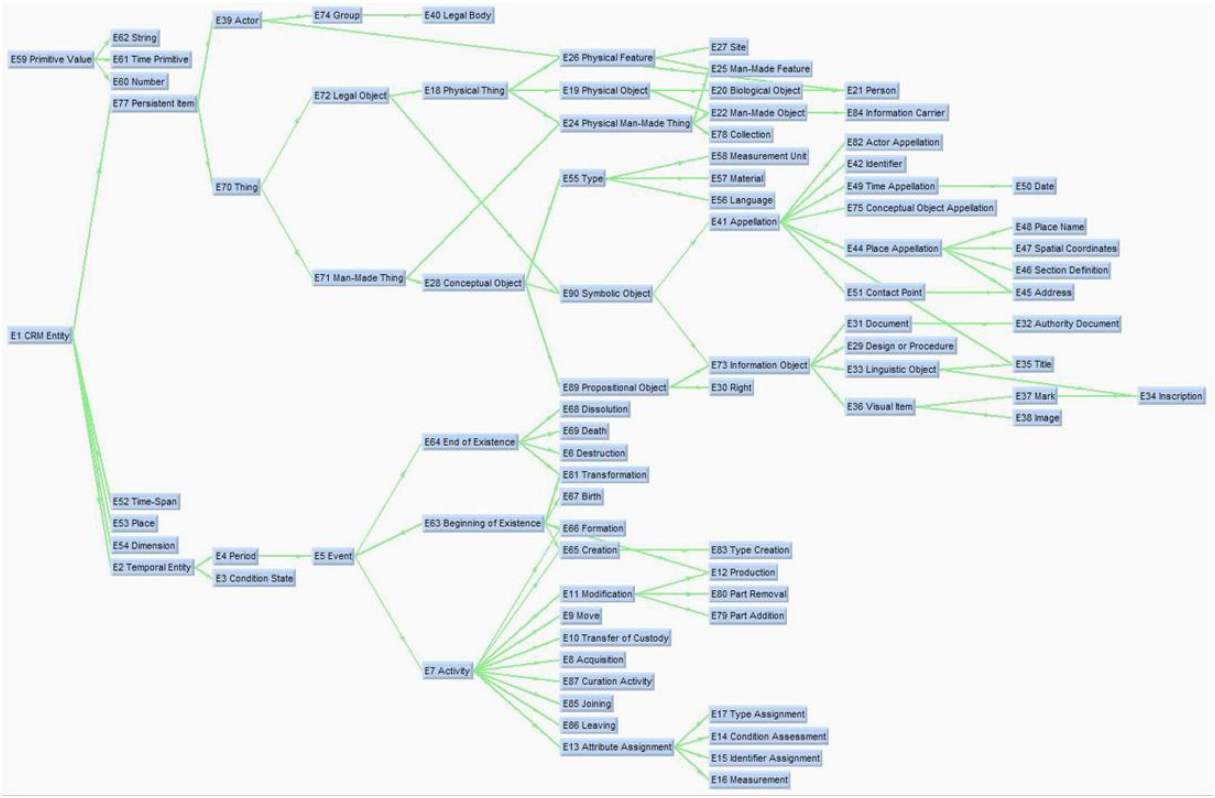

ABBILDUNG 11. CIDOC CRM KLASSEN-HIERARCHIE. QUELLE: [HTTP://WWW.CIDOC-CRM.ORG/CIDOC_GRAPHICAL_REPRESENTATION_V_5_1/CLASS_HIERARCHY.HTML](http://www.cidoc-crm.org/cidoc_graphical_representation_v_5_1/class_hierarchy.html) [14.2.2015].

Die Properties des CRMs werden durch einen mit „P“ beginnenden und mit einer ganzen Zahl folgenden Code identifiziert. Die inverse Property einer Property, d.h. die Eigenschaft, die die umgekehrte Eigenschaft zwischen zwei Entitäten beschreibt, wird durch den Code der ursprünglichen Eigenschaft mit Suffix „i“ bezeichnet. Die natürlichsprachliche Bezeichnung der Properties ist ein transitives Prädikat. Wie Klassen werden Properties nicht formal sondern ebenfalls durch eine *Scope Note* definiert. Zusätzlich formal definiert werden die Properties allerdings durch die Spezifizierung von Ausgangsklasse (Domain) und Zielklasse (Range). Verbindungen von Entitäten durch Properties lehnen sich somit an die Subjekt-Verb-Objekt Struktur vieler Sprachen (insbesondere des Englischen) an. Auch zwischen Properties sind wie im Falle der Klassen im CRM zur semantischen Spezialisierung/Generalisierung hierarchische Abhängigkeiten definiert. Um die objektorientierte Polymorphie in diesen Fällen zu gewährleisten müssen Domain und Range einer Property mit der Domain und Range der Superproperty mindestens übereinstimmen oder Superklasse der Domain und Range der Superproperty sein.

Letztlich sei noch erwähnt, dass im Gegensatz beispielweise zu manchen objektorientierten Programmiersprachen wie beispielsweise Java, im CRM

Mehrfachvererbung, d.h. die Vererbung von Properties mehrerer Klassen an eine Subklasse möglich ist.

Nachdem auf die allgemeine Struktur des CIDOC CRM eigegangen wurde, sollen nun kurz die wichtigsten Konzepte des CIDOC CRM nach [Oldman und C.R.M. Labs 2014] näher beschrieben werden[16].

Die Elemente des CRMs sind Klassen, Properties und Entitäten, d.h. Instanzen von Klassen.

Die Klassen in der Hierarchie des CRM sind alle Subklassen der Klasse *E1 CRM Entity* (*E1 CRM Entität*), die als Wurzel des Hierarchiebaumes die Klasse aller instanziierbaren Klassen darstellt. Außerhalb dieses in *E1* wurzelnden Baumes existiert die Klasse *E59 Primitive Value* (*E59 Primitiver Wert*) und ihre Unterklassen, die unstrukturierte Daten aufnehmen.

Die Klasse *E1* aller CRM Entitäten teilt sich weiter in zwei Unterklassen auf, nämlich *E2 Temporal Entity* (*E2 Geschehendes*) und *E77 Persistent Item* (*E77 Seiendes*). Erstere Klasse enthält zeitlich beschränkte Entitäten wie Ereignisse und Aktionen, letztere Klasse enthält Dinge, die für eine unbestimmte Zeit existieren wie Menschen, Objekte oder auch Ideen. Beide Klassen sind disjunkt, d.h. jede Entität des CRM (außer *E1 CRM Entity*) ist entweder eine Entität der einen oder der anderen Klasse, niemals allerdings beider.

Einige abstraktere Klassen der Klassenhierarchie, die besonders zentrale Konzepte des CRMs beschreiben, sollen im Folgenden kurz vorgestellt werden.

Die zentrale Klasse des ereigniszentrierten CRMs ist die Klasse *E2 Temporal Entity* (*E2 Geschehendes*). Agenten der Klasse *E39 Actor* (*E39 Akteur*) sind an Ereignissen der Klasse *E2 Temporal Entity* beteiligt, die wiederum materielle Dinge der Klasse *E18 Physical Thing* (*E18 Materielles*) oder konzeptuelle Objekte der Klasse *E28 Conceptual Object* (*E28 Begrifflicher Gegenstand*) betreffen. Ereignisse finden während einer gewissen Zeitspanne *E52 Time Span* (*E52 Zeitspanne*) an einem bestimmten Ort *E53 Place* (*E53 Ort*) statt. Typen der Klasse *E55 Type* (*E55 Typus*) erlauben das Klassifizieren beispielsweise von Entitäten der Klassen *E39 Actor*, *E18 Physical Thing* oder *E28 Conceptual Object*. *E41 Appellation* (*E41 Benennung*) schließlich erlaubt das

[16] Hier wie auch im Folgenden werden für die deutschen Namen der Elemente des CIDO CRM die offizielle deutsche Übersetzung verwendet: Definition des CIDOC Conceptual Reference Model: Version 5.0. 1., autor. durch die CIDOC CRM Special Interest Group (SIG). ICOM Deutschland, 2010. Verfügbar unter http://www.icom-deutschland.de/client/media/380/cidoccrm_end.pdf [08.05.2015].

Identifizieren von Entitäten. Fast alle anderen Klassen sind Subklassen der gerade vorgestellten Top-Level-Klassen.

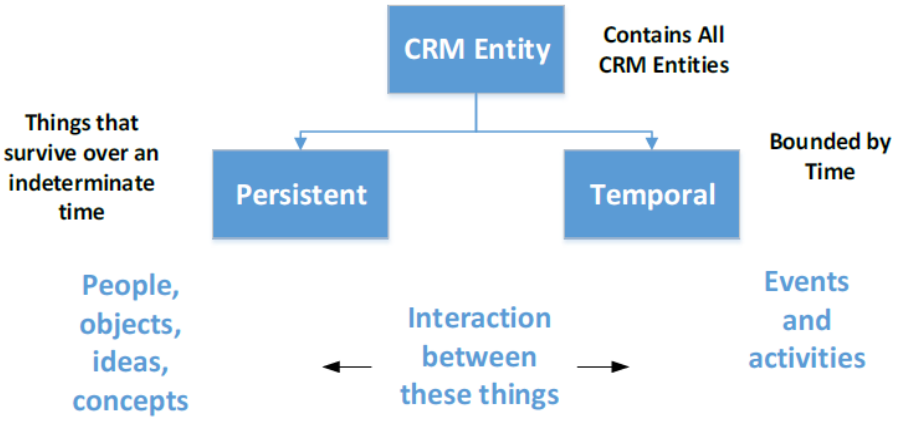

ABBILDUNG 12. PERSISTENTE UND TEMPORÄRE ENTITÄTEN. QUELLE: OLDMAN, D., & LABS, C. R. M. (2014). THE CIDOC CONCEPTUAL REFERENCE MODEL (CIDOC-CRM): PRIMER, SEITE 8.

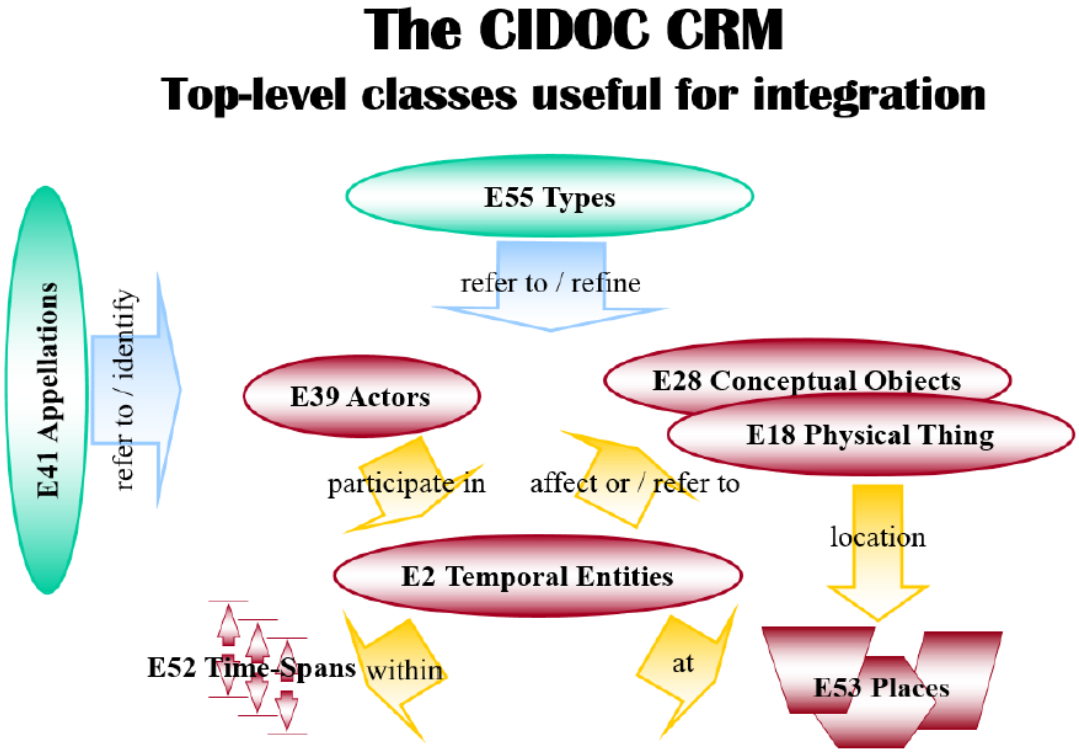

ABBILDUNG 13. CRM TOP-LEVEL KLASSEN. QUELLE: OLDMAN, D., & LABS, C. R. M. (2014). THE CIDOC CONCEPTUAL REFERENCE MODEL (CIDOC-CRM): PRIMER, SEITE 10.

Schließlich sollen noch auf sogenannte Shortcuts, d.h. Abkürzungen, unter den Properties des CIDOC CRM aufmerksam gemacht werden. Diese stellen eine Abkürzung eines kompletten, expliziten Weges über mehrere Properties und Klassen in Form einer einzigen Property dar. So stellt beispielweise die Property *P43 has dimension* (*P43 hat Dimension*) von der Domain *E70 Thing* (*E70 Sache*) zur Range *E54 Dimension* (*E54 Maß*) einen Shortcut des kompletten Weges von *E70 Thing* über die Property *P39i was measured by (P39i wurde vermessen durch*) zu einer Entität der Klasse *E16 Measurement* (*E16 Messung*) und dann von dort weiter über die Property *P40 observed dimension* (*P40 beobachtete Dimension*) zu *E54 Dimension* dar. Zwar ist die Verwendung von Shortcuts bequem, allerdings können so intermediäre Entitäten des kompletten Weges nicht näher beschrieben werden. Es ist daher von Fall zu Fall zu entscheiden, ob auf einen Shortcut zurückgegriffen werden kann oder nicht.

3 Forschungsstand

3.1 Schema-Ontologie-Mapping

Dank dem Fortschritt von Informations- und Kommunikationstechnologien und der zunehmenden Verbreitung ihrer Anwendung sind enorme, allerdings auch disparate digitale Datenmengen angewachsen. Wegen der Heterogenität dieser Daten - sowohl syntaktisch, strukturell als auch semantisch -, deren Verteiltheit und dem Wunsch nach der interoperablen automatischen Verarbeitung dieser Datenmengen, spielen integrative Technologien daher eine zunehmende Bedeutung.

Während zur Lösung der Probleme syntaktischer und struktureller Heterogenität von Daten bereits etablierte Ansätze bestehen, ist das Problem semantischer Heterogenität, die durch variierende Bedeutung und ambige Interpretationen der Daten entsteht, noch nicht gelöst.

Datenbanken mit relationalen Schemata sind eine seit langem weit verbreitete Technologie, um Daten zu strukturieren und zu verwalten. Doch vor allem in den letzten Jahren sind zu diesem Zweck im Kontext des Semantic Webs Ontologien zunehmend bekannter geworden. Die Semantic Web-Bewegung hat es sich dabei zum Ziel gemacht die Transition von einem Web der Dokumente zu einem Web der Daten zu realisieren. Das Semantic Web stellt dabei selbst ein globales Web verlinkter Daten, d.h. eine enorme globale Datenbank, dar.

Die Welt der Datenbanken und die Welt des Semantic Webs sind derzeit allerdings noch zwei unterschiedliche Welten. Zur Überwindung dieser Kluft bedarf es der Transformation zwischen diesen beiden Welten.

Die Transformation zwischen Datenbank und Ontologie kann von verschiedenen Motiven getrieben sein [Spanos, Stavrou und Mitrou 2012]:

- Die semantische Annotation dynamischer Webseiten
- Integration heterogener Datenbanken
- Ontologie-basierte Integration bei denen die Ontologien als globales konzeptuelles Schema dienen
- Ontologie-basierter Datenzugriff (OBDA - Ontologie-Based Data Access)
- Massengenerierung von Semantic-Web-Daten
- Ontologie-Lernen
- Definition der intendierten Bedeutung von Datenbankschemata
- Integration von Datenbankinhalten mit anderen Datenquellen
- Realisierung des Linked-Data-Paradigmas, d.h. unter Anderem der Verwendung verbreiteter Vokabulare (z.B. Ontologien), der Verlinkung der

Datensätze und der Verwendung von Identifikatoren, die realweltliche Entitäten beschreiben

Eine Technologie zur Transformation zwischen relationalen Daten und Ontologien, das Schema/Ontologie-Mapping, soll im Folgenden näher vorgestellt werden.

Die Anzahl der zu diesem Thema in den letzten Jahren publizierten Artikel ist bemerkenswert, ja sogar überwältigend, so Otero-Cerdeira, Rodíguez-Martínez und Gómez-Rodríguez [Otero-Cerdeira, Rodíguez-Martínez und Gómez-Rodríguez 2015]. Mehrere Artikel der letzten Jahre bieten daher Überblicksdarstellungen zum Thema mit Rezensionen zu Literatur und Mapping-Tools (beispielsweise [Otero-Cerdeira, Rodíguez-Martínez und Gómez-Rodríguez 2015], [Shvaiko und Euzenat 2013], [Choi, Song und Han 2006], [Doan und Halevy 2005], [Kalfoglou und Schorlemmer 2003]).

Das Mapping ist eine Operation, die semantisch entsprechende Entitäten unterschiedlicher Schemata oder Ontologien einander zuordnet. Das Ergebnis dieser Operation ist ein sogenanntes Alignement, d.h. eine Menge von Entitätspaaren oder Korrespondenzen, die semantisch entsprechende Entitäten aus beiden Schemata/Ontologien darstellen (Abbildung 14).

Zusätzlich können weitere Parameter am Mapping-Prozess beteiligt sein. Insbesondere ist hier die Verwendung bereits bestehender Alignements, vom Mapping-Prozess verwandter Parameter wie Schwellwerte oder Gewichtungen oder auch externer Ressourcen wie beispielsweise Wörterbücher oder Thesauri zu nennen.

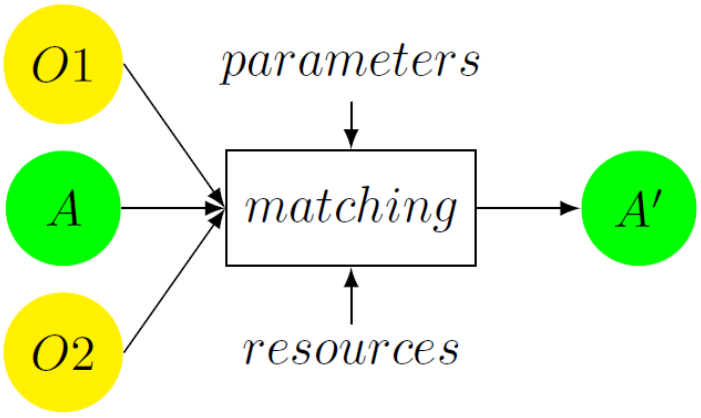

Abbildung 14. Schematische Darstellung des Matching-Prozesses. Quelle: Shvaiko, P., & Euzenat, J. (2013). Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, *25*(1), 158-176, Seite 159.

Spanos, Stavrou und Mitrou [Spanos, Stavrou und Mitrou 2012] geben einen Überblick über verschiedene Ansätze des Schema/Ontologie-Mappings und klassifizieren diese (Abbildung 15).

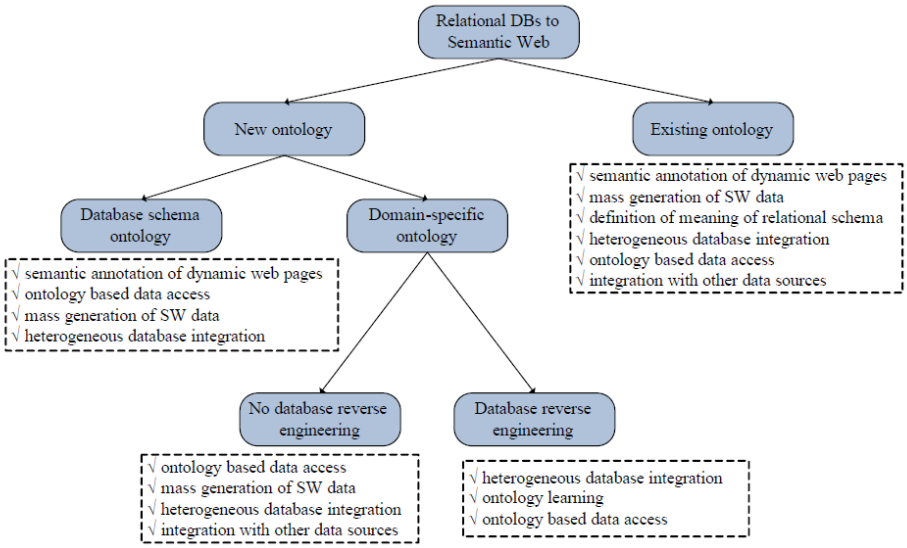

Abbildung 15. Klassifikation von Ansätzen des Schema/Ontologie-Mappings, Quelle: Spanos, D. E., Stavrou, P., & Mitrou, N. (2012). Bringing relational databases into the semantic web A survey. Semantic Web, 3(2), 169-209, Seite 178.

Die Ansätze des Schema/Ontologie-Mappings lassen sich in zwei Hauptklassen einteilen: Zum einen jene Methoden, die aus den Datenbanken eine neue Ontologie ableiten, und zum anderen jene, die Mappings zwischen Datenbank und einer (oder mehrerer) bereits existierender Ontologien aufdecken und definieren.

In letzterem Fall kann weiter nach der Art der Domäne der zu erstellenden Ontologie unterschieden werden. Die Domäne der Ontologie kann dabei das relationale Modell selbst sein, man spricht hier von einer Datenbankschema-Ontologie, oder aber eine andere spezifische Domäne modellieren. In letzterem Fall kann man schließlich noch jene Methoden gesondert zusammenfassen, die speziell darauf abzielen das ursprüngliche konzeptuelle Modell der Datenbank wiederherzustellen (*database reverse engineering*) und in eine Ontologie zu übersetzen.

Die Ansätze lassen sich weiter durch verschiedene Parameter beschreiben. Hier ist zunächst einmal der Grad der Automation des Datenbank-Ontologie Mappings ein Mittel der Klassifizierung. Hierbei reichen die Manifestationen von einem automatischen, über ein semi-automatisches bis zu einem rein manuellen Mapping.

Auch die Art des Datenzugriffparadigmas erlaubt eine Unterscheidung der Ansätze. Das *ETL*-Paradigma (Extract, Transform, Load) auch *batch transformation* oder

massive dump genannt stellt ein materialisiertes Mapping dar bei dem die Daten in der Datenbank repliziert werden und beispielsweise in einem Triple-Store, d.h. einer speziellen Datenbank für die Speicherung von RDF-Tripeln, gespeichert werden. Beim virtuellen Mapping, auch *query driven* oder *on demand*, genannt, werden nicht alle relationalen Daten gleich repliziert sondern erst bei Anfrage - zum Beispiel über SPARQL, einer Anfragesprache und Protokoll das auf dem RDF-Modell operiert - die relevanten Daten in das Ontologiemodell transformiert.

Des Weiteren lassen sich die Mappings beispielsweise durch die verwendete Mapping-Sprache, die verwendete Ontologiesprache der am Mapping beteiligten Ontologien, die Wiederverwendung von vordefinierten Vokabularen - einer Voraussetzung einer erfolgreichen Semantic Web-Realisierung - sowie durch das Vorhandensein einer graphischen Benutzerschnittstelle und Zielsetzung des Mappings charakterisieren.

Mapping-Tools, die auf Ansätzen des Mappings von relationalen Daten auf eine bereits existierende Ontologie basieren, realisieren meist entweder die Definition des Mappings oder die Ontologiepopulation, d.h. das Überführen der relationalen Daten auf Grundlage des zuvor definierten Mappings in die Ontologie (Abbildung 16).

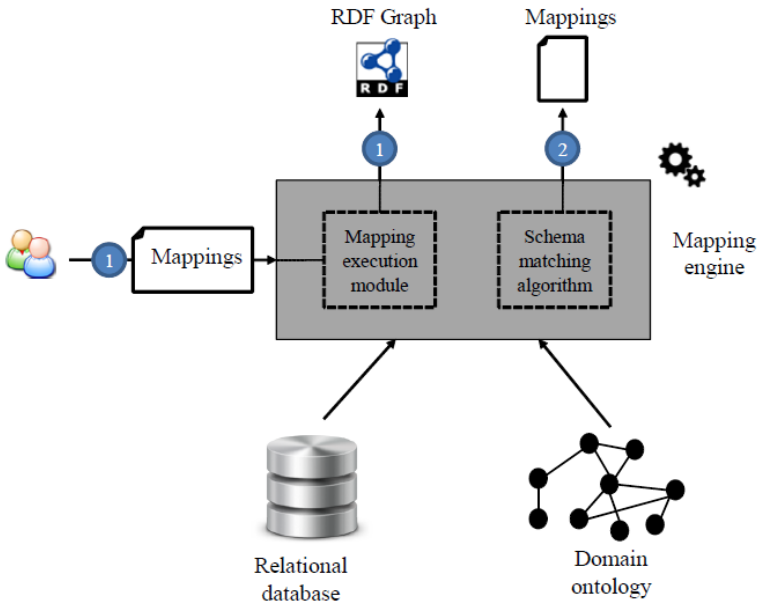

ABBILDUNG 16. MAPPING EINER RELATIONALEN DATENBANK AUF EINE EXISTIERENDE ONTOLOGIE, QUELLE: SPANOS, D. E., STAVROU, P., & MITROU, N. (2012). BRINGING RELATIONAL DATABASES INTO THE SEMANTIC WEB: A SURVEY. *SEMANTIC WEB, 3*(2), 169-209, SEITE 195.

Ghawi und Cullot [Ghawi und Cullot 2007] stellen die Architektur eines Systems für das Datenbank-Ontologie-Mapping mit dem Ziel der semantischen Interoperabilität zwischen heterogenen Informationssystemen vor.

Sie unterscheiden zwischen drei ontologiebasierten Ansätzen der Informationsintegration. Zum einen gibt es die Möglichkeit alle Informationsquellen auf eine globale Ontologie abzubilden. Zum andern kann aber auch jede Informationsquelle durch eine eigene Ontologie beschrieben werden. Für jede dieser lokalen Ontologien muss dann ein Mapping zu allen anderen lokalen Ontologien erstellt werden. Ein hybrider Ansatz aus den beiden zuvor genannten Ansätzen besteht in der Beschreibung der Informationsquellen durch lokale Ontologien und die zusätzliche Verwendung einer globalen Ontologie, die den Anspruch hat die Semantik der gesamten betroffenen Wissensdomäne zu beschreiben. Bei diesem hybriden Ansatz müssen daher zum einen Mappings zwischen den lokalen Informationsquellen und den lokalen Ontologien, als auch zwischen den lokalen Ontologien und der globalen Ontologie erstellt werden.

Auf das konkrete Mapping mit der Ontologie des CIDOC CRM gehen Binding, May und Tudhope [Binding, May und Tudhope 2008] oder Tudhope, Binding, May und Charno [Tudhope, Binding, May, Charno 2013] ein.

In vielen Anwendungen wird das Mapping manuell von Domänenexperten - eventuell unterstützt von graphischen Benutzerschnittstellen - durchgeführt. Diese Tätigkeit ist in der Regel mit einem hohen, zeitintensiven Arbeitsaufwand verbunden. Eine automatisierte Unterstützung des Mappings kann hier Abhilfe schaffen. Zu diesem Zweck wurden verschiedene automatische Ansätze entwickelt[17].

3.2 Mapping-Prozess: Ansätze

Hier soll wie [Shvaiko 2005; Shvaiko 2013] nicht strikt zwischen relationalem und ontologischem Mapping unterschieden werden. Dies hat zwei Gründe: Zum einen sind die Datenmodelle von Datenbankschemata und Ontologien insofern ähnlich als

[17] In einem nicht-experimentellen, praktischen Anwendungskontext ist die vollständige Automatisierung des Mappings allerdings derzeit in den meisten Fälle nicht denkbar und Bedarf letzten Endes doch die Validierung durch den menschlichen Domänenexperten.

sie beide Vokabulare bereitstellen und die Bedeutung der Begriffe des Vokabulars einschränken, wodurch Techniken die für Mapping-Probleme eines dieser beiden Datenmodelle entwickelt wurden oft auch für das andere nützlich sind, zum anderen sind die im Rahmen dieser Arbeit mit der CIDOC CRM-Ontologie zu mappenden in einer relationalen Datenbank vorgehaltenen Daten bereits auf konzeptueller Ebene ebenfalls als graphbasierte Ontologie angelegt (siehe Kapitel 2.1.1). Im Folgenden soll daher im Kontext automatischer Matching-Ansätze nicht zwischen spezifisch relationalen bzw. ontologischen Ansätzen unterschieden werden.

[Rahm und Bernstein 2001] unterscheiden in ihrer Klassifikation automatischer Matching-Ansätze zunächst, ob es sich bei den Matching-Ansätzen um individuelle Ansätze, d.h. Ansätze, die auf einem einzigen Kriterium beruhen, oder um kombinierte Ansätze, die mehrere individuelle Matching-Ansätze verbinden, handelt. Letztere können wiederum in hybride Ansätze, d.h. Ansätze, die auf mehreren Kriterien beruhen, und komposite Ansätze, d.h. Ansätze, die Ergebnisse mehrerer individueller Algorithmen verarbeiten, unterschieden werden.

Die individuellen Matching-Ansätze lassen sich in einem ersten Schritt weiter danach unterscheiden, ob sie Schema-Informationen oder Instanz-Informationen, d.h. die formale Beschreibung der Struktur der Daten oder deren eigentliche Inhalte, verarbeiten. Ansätze beider Klassen können des Weiteren auf Element-Ebene operieren, d.h. auf der Ebene individueller Elemente, im Falle von Schema-basierten Ansätzen aber auch auf Strukturebene, wobei hier ganze Strukturen von individuellen Elementen mit ihren Verbindungen zueinander verarbeitet werden.

Diese Klassen von Matching-Ansätzen schließlich können auf Elementebene weiter in linguistische und Constraint-basierte Ansätze unterteilt werden. Auf Strukturebene kommen in erster Linie Constraint-basierte Ansätze zum Einsatz. Linguistische Ansätze basieren auf der Verarbeitung von Text und Sprache, z.B. in Form von Namen und textuellen Beschreibungen von Elementen. Constraint-basierte Ansätze nutzen die strukturellen Einschränkungen von individuellen Elementen oder ganzer Strukturen aus Elementen.

Shvaiko und Euzenat [Shvaiko und Euzenat 2005] verfeinern diese Klassifizierung für individuelle, schemabasierte Matching-Ansätze. Bei den auf der Verarbeitung von Schema-Informationen basierenden automatischen Matching-Ansätzen lassen sich die unten genauer beschriebenen Klassen nach elementaren Techniken unterscheiden (Abbildung 17).

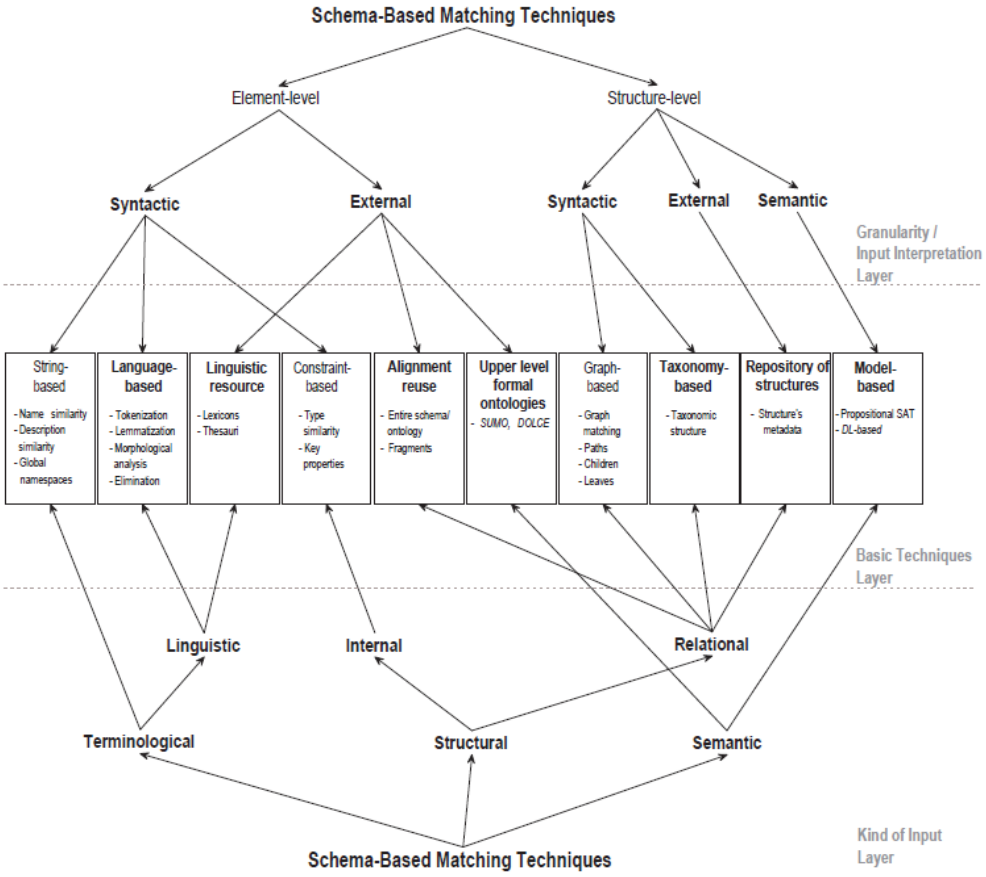

ABBILDUNG 17. KLASSIFIKATION ELEMENTARER SCHEMA-BASIERTER MAPPING-ANSÄTZE, QUELLE: SHVAIKO, P., & EUZENAT, J. (2005). A SURVEY OF SCHEMA-BASED MATCHING APPROACHES. IN *JOURNAL ON DATA SEMANTICS IV* (PP. 146-171). SPRINGER BERLIN HEIDELBERG, SEITE 156.

Ihre Klassifizierung nach Granularität bzw. Interpretation der Eingangsdatenstrukturen unterscheidet zunächst zwischen Mapping-Techniken auf Element-Ebene und solchen auf Struktur-Ebene. Erstere operieren nur auf den isolierten Elementen der Eingangsdatenstrukturen und ignorieren deren Beziehungen zu anderen Elementen. Techniken auf Struktur-Ebene operieren im Gegensatz dazu auf der Struktur der miteinander verbundenen Elemente.

Weiter wird zwischen syntaktischen, externen und semantischen Techniken unterschieden. Syntaktische Techniken interpretieren dabei ausschließlich die interne Struktur der Eingangsdatenstrukturen, während externe Techniken externe Ressourcen, die über die Eingangsdatenstrukturen hinausgehen, mit in die Verarbeitung einbeziehen. Diese externen Ressourcen können beispielsweise Thesauri oder auch menschliche Eingaben sein. Semantische Techniken haben die

Besonderheit bei der Verarbeitung der Eingangsdatenstrukturen auf modelltheoretische, formale Semantiken zurückzugreifen. Da die Semantik in der Verknüpfung von Entitäten liegt, existiert diese Unterklasse nur für Techniken auf Strukturebene.

Das zweite Klassifikationskriterium von Shvaiko und Euzenat, die Art der Eingangsdatenstrukturen, teilt die Ansätze in folgende Klassen: Auf einer ersten Ebene wird zwischen terminologischen, d.h. auf Zeichenketten operierenden Techniken, strukturalen, d.h. auf der Struktur der Eingangsdatenstrukturen operierenden, und semantischen Techniken unterschieden. Letztere benutzen semantische Technologien, um Korrespondenzen zwischen Entitäten abzuleiten.

Auf der zweiten Ebene werden die gerade genannten Klassen weiter verfeinert. Terminologische Techniken können demnach danach unterschieden werden, ob die Zeichenketten als rein solche oder als linguistische Objekte interpretierte werden. Bei den strukturalen Methoden wird weiter zwischen internen Techniken, d.h. Techniken, die auf der internen Struktur von Entitäten operieren, und relationalen Techniken, die auf den Beziehungen zwischen Entitäten operieren, unterschieden.

Die Klassen von elementaren Matching -Techniken, die sich nach obiger Klassifikation ergeben, sollen im Folgenden nach Art der zum Matching verwendeten Informationen näher beschrieben werden.

Terminologische Matching-Ansätze benutzen Wörter und Text, um ähnliche Elemente in den zu mappenden Schemata bzw. Ontologien zu finden. Diese Ansätze betrachten meist die Namen der zu mappenden Elemente oder zusätzliche Kommentare in natürlicher Sprache, die die Elemente näher beschreiben.

Rein zeichenkettenbasierte Techniken werden meist benutzt, um mithilfe von Namen oder Beschreibungen von Entitäten diese aufeinander abzubilden. Die Idee dabei ist, dass Entitäten mit ähnlichen Namen auch ähnliche Konzepte beschreiben. Cohen, Ravikumar, und Fienberg stellen in [Cohen, Ravikumar, und Fienberg, 2003] von dieser Art Ansätzen verwendete Metriken vor. Dabei werden in der Regel Distanzfunktionen verwendet, die Paaren von Zeichenketten reelle Zahlen (meist normalisiert auf ein Intervall zwischen 0 und 1) zuweisen, die die Ähnlichkeit dieser Zeichenkettenpaare ausdrücken sollen. Eine wichtige Klasse dieser Distanzfunktionen sind sogenannte Edit-Distanz-Funktionen, die die Kosten von Edit-Operationen berechnen, die zum Umwandeln einer Zeichenkette in eine andere notwendig ist. Zwei häufig von Edit-Distanzen-Funktionen verwendete Metriken sind die Levenstein-Metrik und die Monger-Elkan-Metrik.

Bei sprachbasierten Techniken werden Zeichenketten als linguistische Objekte interpretiert. Durch Techniken der Verarbeitung natürlicher Sprache, können die Zeichenketten analysiert werden. Häufig dazu eingesetzte Techniken sind beispielsweise die Zerlegung der Zeichenketten in Wörter, die Lemmatisierung von Wörtern, d.h. das Zurückführen auf Grundformen, das Entfernen von Stoppwörtern und die morphologische Analyse von Wörtern und Sätzen.

Techniken, die auf linguistische Ressourcen zurückgreifen, verwenden meist zusätzlich allgemeinsprachliche oder domänenspezifische Thesauri, um Wörter auf Grundlage ihrer linguistischen Beziehungen zueinander (z.B. Synonyme, Hyper-, und Hyponyme) miteinander in Verbindung zu setzen. Ein bekannter Vertreter eines allgemeinsprachlichen Thesaurus der englischen Sprache, der Synonyme, sogenannte SynSets, verzeichnet, ist WordNet[18].

Strukturelle Matching-Ansätze operieren auf den strukturellen Einschränkungen und Abhängigkeiten der zu mappenden Elemente.

Constraint-basierte Techniken operieren auf internen Beschränkungen von zu mappenden Elementen. Dazu gehören beispielsweise Datentypen oder Kardinalitäten von Beziehungen zu anderen Elementen. Oft führen diese Art Ansätze nicht zu Ähnlichkeitswerten sondern zu imperfekten n:m-Mappings (sogenannte Clusters), da mehrere Elemente in diesen Fällen meist für ein Mapping in Frage kommen. Dennoch sind sie nützlich die Menge möglicher Korrelationen zwischen zu mappenden Elementen einzuschränken.

Die Klasse graphbasierter Techniken verwendet Graphalgorithmen und interpretiert die Eingangsdatenstrukturen nicht als Menge von Einzelelementen sondern als zusammenhängende Graphen. In der Regel beruhen dabei die Ähnlichkeitsvergleiche zwischen zwei Knoten auf deren Position im Graph. Dieser Art Techniken liegt die Annahme zu Grunde, dass, wenn zwei Knoten in zwei Ontologien ähnlich sind, auch deren Nachbarn ähnlich sind. Es handelt sich dabei meist um Optimierungsprobleme, die durch einen Fixpunkt-Algorithmus, d.h. ein Algorithmus, der schrittweise näherungsweise Lösungen verbessert, bis keine Verbesserung mehr erreicht wird, gelöst werden. Ein solcher Fixpunkt-Algorithmus ist beispielsweise der von Melnik, Garcia-Molina und Rahm in [Melnik, Garcia-Molina und Rahm 2012] beschriebene *Similarity Flooding*-Algorithmus, der zwei Graphen als Eingabe verarbeitet und ein Mapping zwischen korrespondierenden Knoten des Graphen zurückliefert. Zunächst werden die zu mappenden Datenmodelle in einen gerichteten und bezeichneten Graphen überführt. Dann werden durch eine iterative Fixpunkt-Berechnung

[18] http://wordnet.princeton.edu/ [04.02.2015].

Ähnlichkeitswerte zwischen Knoten beider Graphen berechnet. Dabei sind zwei Knoten in den jeweiligen Graphen umso ähnlicher, je ähnlicher ihre Nachbarknoten sind. Ähnlichkeiten propagieren sich somit durch die Graphen. Ein anderer graphbasierter Algorithmus wird von Hu, Jian, Qu und Wang [Hu, Jian, Qu und Wang 2005] vorgeschlagen. Ihr *GMO* genannter Ansatz soll den Vorteil haben im Gegensatz zu anderen graphbasierten Ansätzen, die selbst auf weiteren, insbesondere lexikalischen Ansätzen, beruhen, allein auf der Struktur von Graphen zu operieren.

Eine speziellere Klasse von Techniken, die ebenfalls auf Graphalgorithmen zurückgreifen sind taxonomiebasierte Techniken**.** Hier werden allerdings nur Generalisierungs- und Spezialisierungsbeziehungen zwischen Elementen der zu mappenden Datenstrukturen betrachtet.

Manche strukturelle Techniken greifen auf externe Ressourcen zurück. So können Alignments bereits zuvor gemappter Schemata bzw. Ontologien wiederverwendet werden. Diesen Techniken liegt die Annahme zugrunde, dass bereits erstellte Mappings in derselben Anwendungsdomäne neu zu erstellenden Mappings ähnlich sind.

Andere externe Ressourcen verwendende Techniken benutzen Struktur-Repositorien, die Schemata bzw. Ontologien und deren Fragmente zusammen mit deren paarweisen Ähnlichkeitswerten vorhalten. Im Gegensatz zu wiederverwendeten Alignements speichern Struktur-Repositorien also nur Ähnlichkeitswerte zwischen Schemata bzw. Ontologien und nicht die Korrespondenzen selbst.

Semantische Matching-Ansätze werden nach [Shvaiko und Euzenat 2005] bis dato nur wenig genutzt.

Modellbasierte Techniken verarbeiten hier Eingangsdatenstrukturen aufgrund ihrer semantischen Informationen.

Die Hinzunahme von Upper-Level-Ontologien kann bei der Interpretation der zu mappenden Ontologien durch das zusätzlich durch sie bereitgestellte Wissen dienen.

4 Lösungsansatz

4.1 Diskussion

Eine vollständige Automatisierung des Mapping-Prozesses soll und kann im Rahmen dieser Arbeit nicht geleistet werden. Gegen die vollständige Automatisierung sprechen zwei Gründe:

- Zum einen ist die Ontologie des CIDOC CRM äußerst komplex und zugleich sehr abstrakt. Der Versuch eines vollständig automatischen Mappings führt aus diesem Grund aller Voraussicht nach nur zu Mappings mit geringer Präzision.
- Zum anderen ist das zu realisierende Mapping-Tool nicht für den experimentellen sondern für den praktischen Einsatz gedacht. Für diesen ist die Erstellung eines präzisen Mappings allerdings essentiell.

Der Beitrag des Domänenexperten ist für ein verlässliches Mapping daher unerlässlich. Das zu entwickelnde Tool soll deswegen nicht als vollautomatisches sondern als den Mapping-Prozess strukturierendes und den Domänenexperten durch die Mapping-Aufgabe führendes, semi-automatisches, unterstützendes Werkzeug mit einer graphischen Benutzerschnittstelle konzipiert werden.

Ein großes Problem mit dem sich der Domänenexperte beim Mapping eines Quelldatenmodelles auf das Zieldatenmodell des CIDOC CRM konfrontiert sieht, ist die große Freiheit in der Interpretation der Konzepte, die letzteres zulässt. Der CIDOC-CRM-Standard selbst definiert ausschließlich ein Modell auf konzeptueller Ebene und gibt keine Richtlinien für dessen tatsächliche Implementierung in einer Anwendung.

Nußbaumer, Haslhofer und Klas entwickeln daher eine Mapping-Methode [Nußbaumer, Haslhofer und Klas 2010] zur Modellimplementierung des CIDOC CRM. Ihr erklärtes Ziel ist es dabei die Divergenz von Mappings zwischen Quelldatenmodellen und dem CRM als Zieldatenmodell zu reduzieren und damit die Interoperabilität der aus verschiedenen Mappings mit dem CRM resultierenden Daten zu stärken.

Ihr Vorgehen besteht dabei aus drei Schritten. Zunächst einmal wird das Quelldatenmodell in dieselbe technische Repräsentation wie das CRM überführt. Daran anschließend werden die Konzepte des Quelldatenmodells mit den Konzepten des CRM gemappt. Dabei müssen semantisch korrespondierenden Konzepte in Quellmodell und CRM entdeckt und für die weitere Verarbeitung in einer geeigneten

Datenstruktur repräsentiert werden. Schließlich wird das aus der Mapping-Operation resultierende Alignment bei der Datenintegration auf Anwendungsebene angewendet.

Die für die ersten beiden Schritte von Nußbaumer et al. entwickelte Methode soll als Ausgangspunkt für die Realisierung des im Rahmen dieser Arbeit zu realisierenden anwendungsspezifischen Tools dienen und soll im Folgenden kurz vorgestellt werden.

Nachdem Quelldatenmodell $Msrc$ und CIDOC CRM $Mcrm$ in eine gemeinsame technische Repräsentation überführt worden sind, sollen die elementaren Instanzen (Entitäten $e0, e1, \dots, en \in Esrc$ und Properties $p0, p1, \dots, pn \in Psrc$) des Quellmodells $Msrc$ auf Instanzen ($ex \in Ecrm$ und $ey \in Pcrm$) des CRM abgebildet werden. Insbesondere sollen die Quelldomains und -ranges von Properties aus $Msrc$ mit Zieldomains und -ranges aus $Mcrm$ gemappt werden und ein Zielpfad zwischen Zieldomain und Zielrange aus $Mcrm$ gefunden werden.

Folgende Funktionen werden dazu benötigt:

- isA: $E \times E \rightarrow BOOLEAN$ ist definiert als $isA(ex, ey) = (true|false)$ und liefert true zurück, wenn ex = ey oder eine direkte oder indirekte Subklasse von ey ist
- $getDomain$: $P \rightarrow P(E)$ ist definiert als $getDomain(p) = \{ei \in E\}$ und liefert die Menge aller direkten und indirekten Subklassen der Klasse, die als Domain von p definiert ist, sowie die Klasse selbst zurück
- $getRange$: $P \rightarrow P(E)$ ist definiert als $getRange(p) = \{ei \in E\}$ und liefert die Menge aller direkten und indirekten Subklassen der Klasse, die als Range von p definiert ist, sowie die Klasse selbst zurück

Unter Zuhilfenahme dieser Funktionen werden die Eingangsdatenmodelle verarbeitet und als Ausgabe eine Menge von Korrelationen zwischen Properties des Quelldatenmodells und Properties bzw. Property-Pfaden des CRM erzeugt. Der Ablauf der Mapping-Methode nach Nußbaum et al. ist in Abbildung 18 dargestellt.

Mapping Rule 1: CIDOC CRM mapping methodology

Data: a source specific model $M^{src} \in \mathcal{M}$ and the CIDOC CRM $M^{crm} \in \mathcal{M}$

Result: a set of mapping chains $c_0, c_1, \ldots, c_n \in C$ denoting the semantic correspondences between M^{src} and M^{crm}

1 **forall the** $e \in M^{src}$ *being source domain entities* **do**
2 find the most specific entity $e_{start} \in M^{crm}$describing the semantics of e;
3 define e as instance of e_{start};
4 **forall the** $p \in M^{src}$ *with* $getDomain(p) = e$ **do**
5 $c \longleftarrow \emptyset$;
6 find the most specific entity $e_{end} \in M^{crm}$ describing the semantics of p;
7 define the instance of the range of p as instances of e_{end};
8 add e_{end} to the mapping chain c;
9 $x \longleftarrow e_{end}$;
10 **repeat**
11 find the most specific property $p_{crm} \in M^{crm}$ such that $getRange(p_{crm}) = x$;
12 add p_{crm} to the mapping chain c;
13 find the most specific entity $e_{crm} \in M^{crm}$ such that $getDomain(p_{crm}) = e_{crm}$;
14 add e_{crm} to the mapping chain c;
15 $x \longleftarrow e_{crm}$;
16 **until** $isA(e_{start}, x)$;
17 invert the mapping chain c;
18 $e_{start} \longleftarrow x$;

ABBILDUNG 18. CIDOC CRM MAPPING-METHODE, QUELLE: NUSSBAUMER, P., HASLHOFER, B., & KLAS, W. (2010). TOWARDS MODEL IMPLEMENTATION GUIDELINES FOR THE CIDOC CONCEPTUAL REFERENCE MODEL, SEITE 18.

Für das im Rahmen dieser Arbeit zu realisierende Mapping-Tool soll die oben beschriebene Mapping-Methode an die spezifische Anwendung angepasst und erweitert werden.

Zu diesem Zweck sollen oben vorgestellter Klassifikation von Mapping-Ansätzen folgend im Folgenden diese Mapping-Ansätze im Hinblick auf ihre Anwendbarkeit und erwarteter Effektivität zur Lösung der Problemstellung diskutiert werden.

Da es sich bei der Spezifikation des CIDOC CRMs um Schema-Daten handelt, d.h. Daten die die Struktur der eigentlichen Instanzdaten beschreiben, kann man sich hier auf schemabasierte Mapping-Ansätze beschränken. Diese sollen auf Grundlage des Schemagraphen des CIDOC CRM und des wie oben beschrieben teilweise aus dem

Instanzgraphen abgeleiteten Schemagraphen der ConedaKor-Datenbank zur Unterstützung des Mappings zum Einsatz kommen.

Für die sinnvolle Anwendung terminologischer bzw. linguistischer Ansätze müssen die von diesen Ansätzen zu verarbeitenden Daten der Quell- und Zieldatenstruktur zunächst einmal in einer gemeinsamen Sprache vorliegen. Die in dieser Arbeit verwendete Frankfurter ConedaKor-Datenbank verwendet zur Benennung und Beschreibung ihrer Elemente die deutsche Sprache. Das CRM bietet hier eine gewisse Flexibilität, indem für die Bezeichnung seiner Elemente bereits Standardübersetzungen in vielen verschiedenen Sprachen, darunter das Deutsche, existieren[19]. Allerdings haben die Konzepte des Quellmodells und des CIDOC CRMs einen sehr unterschiedlichen Abstraktheitsgrad. Während das Quellmodell der am kunstgeschichtlichen Institut der Universität in Frankfurt verwendeten ConedaKor-Datenbank, die im Rahmen dieser Arbeit als Quellmodell dient, eher konkrete Konzepte beinhaltet und dementsprechende Bezeichnungen verwendet (z.B. „Literatur“, „Werk“), besteht das CRM aus weitaus abstrakteren Konzepten, die in der Regel dann auch andere, unscharfe Bezeichnungen tragen (z.B. „E73 Information Object“ („E73 Informationsgegenstand“), „E24 Physical Man-Made Thing“ („E24 Hergestelltes“)). Die Hinzunahme externer linguistischer Ressourcen wie Wörterbücher oder Thesauri, die Beziehungen wie Hyper-, Hypo- oder Synonyme zwischen Elementen eines kontrollierten Vokabulars darstellen, könnten hier hilfreich sein.[20] Die Zuhilfenahme eines Thesaurus setzt allerdings die Analyse der Bezeichner voraus, denn diese sind meist (so zumindest im CIDOC CRM) nicht einfache, eigenständige Wörter sondern haben die Struktur einfacher Nominalphrasen. Zur Nutzung eines Wörterbuchs bzw. Thesaurus, der nur Grundformen von Wörtern verzeichnet, müssten also die grammatischen Strukturen der Bezeichner linguistisch analysiert und Wörter auf ihre Grundformen zurückgeführt werden können.

Da sowohl das Datenmodell von ConedaKor als auch das CIDOC CRM interne Strukturen seiner Elemente definiert, können Constraint-basierte Ansätze hier sinnvoll eingesetzt werden, um valide Mappings zu erzeugen, bei denen die Zuordnung strukturell nicht zueinander passender Elemente aus beiden Datenmodelle ausgeschlossen wird. Diese Art Ansätze werden bereits in der

[19] http://www.cidoc-crm.org/translation_guidelines.html [21.02.2015].

[20] Ein der bekannten lexikalischen Datenbank in englischer Sprache *WordNet* (http://wordnet.princeton.edu/ [07.02.2015]) ähnlicher Thesaurus in deutscher Sprache ist beispielsweise *GermaNet* (http://www.sfs.uni-tuebingen.de/GermaNet/ [21.02.2015]) oder *openthesaurus.de* (https://www.openthesaurus.de/ [09.05.2015]).

Mapping-Methode von Nußbaumer et al. angewendet. Hier wird nämlich zur Garantie valider, die Semantik der jeweiligen Datenmodelle nicht verletzender Mappings zwischen den Quellkonzepten und den zu mappende Zielkonzepten aus dem CRM, die Auswahl möglicher Zielkonzepte auf Grundlage von den auf Elementebene definierten Schemainformationen zu Domain und Range von zu mappenden Relations und Properties eingeschränkt.

Existiert keine CRM Property, die alleine bereits einer Kor Relation entspricht, muss diese Relation auf einen Pfad von CRM Properties abgebildet werden, der in seiner Gesamtheit der Semantik der Relation entspricht. Hier können auf Strukturebene, d.h. der Ebene des Netzwerks verknüpfter Elemente**,** graphbasierte Ansätze zum Einsatz kommen. Da der einer Relation aus ConedaKor semantisch entsprechende Pfad zwischen einem Start- und Zielknoten im CRM erfahrungsgemäß auch meistens recht kurz ist, könnte dieser Pfad eventuell über einen Kürzesten-Weg-Algorithmus (Shortest-Path-Algorithmus) vorberechnet werden.

4.2 Zu realisierender Ansatz

Der zu realisierende Lösungsansatz sieht demnach folgendermaßen aus: In einem ersten Schritt sollen die Repräsentationen des CIDOC CRM sowie die Schema-Daten der ConedaKor-Datenbank wie in [Nußbaumer et al. 2010] vorgeschlagen in ein gemeinsames objektorientiertes Datenmodell überführt werden.

Das objektorientierte Datenmodell muss dabei nicht die komplette Logik der CIDOC CRM-Ontologie abbilden, sondern kann sich auf für die Anwendung notwendige logische Abhängigkeiten, insbesondere die Generalisierungs- und Spezialisierungsbeziehungen, zwischen Ressourcen des CRM beschränken.

Die in ein objektorientiertes Datenmodell zu überführenden Schema-Daten der ConedaKor-Datenbank müssen teilweise erst aus den Instanz-Daten der Datenbank abgeleitet werden. Zwar definiert die ConedaKor-Datenbank in ihrem Graphmodell Entitätstypen für jede Entität mit den damit verbundenen strukturellen Einschränkungen, die Relationen der Verknüpfungen zwischen zwei Entitäten können allerdings beliebig gewählt werden, womit die Relationen keine strukturellen Einschränkungen tragen und darüber hinaus auch keine einheitliche Semantik ausdrücken (siehe Kapitel 2.1). Zur Überführung der Schemadaten aus ConedaKor in das zu verarbeitende objektorientierte Datenmodell werden daher aus dem Graph der Dateninstanzen „eigentliche“ Relationen über die Kombination von Relation der im Datengraph existierenden Verknüpfungen und den Entitätstypen deren

verbundener Entitäten abgeleitet. Über diese im konzeptuellen Graphmodell von ConedaKor angelegten Schemainformationen hinaus müssen zusätzlich die nicht konsequent im Graphmodell als Knoten und Kanten modellierten Felder ebenfalls in das objektorientierte Schemadatenmodell überführt werden.

In einem zweiten Schritt soll das eigentliche Mapping zwischen ConedaKor und CIDOC CRM vom Domänenexperten im Dialog über eine die Auswahl semantisch korrespondierender Konzepte unterstützende graphische Benutzerschnittstelle realisiert werden können.

Als zusätzliche Operation zu Nussbaumer et al., deren Methode nur ein Mapping zwischen Relationen vorsieht, soll das zu realisierende Mapping-Tool das Mapping zwischen Kor-Entitätstypen und semantisch entsprechenden Klassen des CRM ermöglichen. Der Benutzer des Tools soll dabei in seiner Arbeit durch die automatische Berechnung der lexikalischen Ähnlichkeit zwischen den Bezeichnern von Kor-Entitätstypen und CRM-Klassen unterstützt werden. Die Klassen des CRMs werden nach berechnetem Ähnlichkeitswert für jeden Entitätstyp geordnet und die Klasse mit dem höchsten Ähnlichkeitswert standardmäßig ausgewählt. Der Domänenexperte kann diese Auswahl validieren oder in der geordneten Liste der anderen CRM-Klassen eine semantisch passendere Klasse auswählen.

Nachdem die Entitätstypen gemappt worden sind, sollen die wie oben beschrieben abgeleiteten „eigentlichen“ Relationen auf eine Property bzw. einen Pfad von Properties und Klassen des CRM abgebildet werden. Als Modifikation zu Nussbaumer et al. sollen die auf die „eigentlichen“ Relationen abzubildenden Pfade aus Properties und Klassen des CRM nicht invertiert, d.h. von der mit der Range der „eigentlichen“ Relation gemappten Klasse des CRM aufgebaut werden. Ein solches Vorgehen scheint weniger intuitiv als der Aufbau von der mit der Domain der „eigentlichen“ Relation gemappten Klasse aus, zudem wird auch keine Begründung für die invertierte Anordnung der Property-Pfade in [Nußbaumer et al. 2010] genannt.

Beim Abbilden von „eigentlichen“ Relationen auf Property-Pfade sollen zur Unterstützung des Mappings durch den Domänenexperten Constraint-basierte Ansätze auf Elementebene genutzt werden, die die Möglichkeiten des Mappings einschränken sollen. Dazu sollen vor allem die Domain- und Range-Einschränkung von „eigentlichen“ Relationen bzw. Properties sowie die Spezialisierungs-/Generalisierungsabhängigkeiten zwischen Klassen dazu genutzt werden valide Verknüpfungen zwischen Klassen und Properties zu erstellen und die dem Domänenexperten zum Mapping angebotene Auswahl an CRM-Ressourcen somit einzuschränken.

Von der Vernetztheit der Schemadaten profitierend, sollen auch graphbasierte Ansätze auf Strukturebene zum Einsatz kommen. Ein besonderes Problem in der Erstellung eines einer „eigentlichen“ Relation entsprechenden Property-Pfades besteht in der Methode von Nußbaumer et al. darin, dass der das Mapping durchführende Domänenexperte die globalen Abhängigkeiten zwischen Klassen und Properties des CRMs genau kennen muss, um einen geeigneten Pfad von Domain- zu Range-Klasse zu erstellen. Da wie bereits erwähnt erfahrungsgemäß häufig kurze Pfade zwischen den gemappten Start- und Zielklassen einer Relation mit dieser korrespondieren, soll versucht werden für jede „eigentliche“ Relation über einen Shortest-Path-Algorithmus ein der Relation semantisch entsprechender Pfad zu berechnen. Da die minimalen Wege von einer Klasse des CRM zu einer anderen nie länger als zwei Properties lang sind und sich die Wege damit nicht so sehr in ihrer Länge als in der Art der sie konstituierenden Properties unterscheiden, muss hier auf weitere Ansätze zurückgegriffen werden, die die Gewichtung der Properties im Graph des CRM und somit die Diskriminierung von Properties ermöglichen.

Entspricht dieser Pfad nach Meinung des das Mapping durchführenden Domänenexperten nicht der Semantik der Relation, oder soll eine intermediäre Entität auf dem Pfad näher beschrieben werden, so soll der Domänenexperte selbst schrittweise die alternierenden Properties und Klassen die den Pfad konstituieren auswählen können. Nach jeder Auswahl einer intermediären Klasse soll erneut der kürzester Weg von dieser Klasse zur Zielklasse berechnet und dem Domänenexperten vorgeschlagen werden.

Nachdem der Mapping-Vorgang beendet wurde soll schließlich als Ausgabe das Alignment zwischen korrespondierenden Konzepten aus ConedaKor und CRM in maschinell lesbarer und weiterverarbeitbarer Form erstellt werden.

5 Umsetzung

Zur Umsetzung des die oben definierten Anforderungen realisierenden Mapping-Tools, das hier auf den Namen *Kor-CRM-Mapper* (*KCM*) getauft werden soll, wurde das auf der dynamischen Skriptsprache Ruby basierende MVC-Framework (Model-View-Controller-Framework) Ruby on Rails (auch einfach Rails oder RoR gegannt) verwendet. Die Wahl dieser Technologie ist darin begründet, dass ConedaKor mittels Rails realisiert ist und das Mapping-Tool so einfach in die bereits bestehende Architektur des ConedaKor-Systems integriert werden kann. Die verwendete Version von Rails ist 4.1.8, die Version der Skriptsprache Ruby selbst 2.1.5p273. Zum Arbeiten mit in RFD/XML kodierten Dokumenten wurden zusätzlich die Ruby-Gems *rdf* und *rdf-xml*[21] verwendet. Zur Realisierung terminologisch/linguistischer Matching-Ansätze kam die Gem *damerau-levenshtein*[22] zum Berechnen von Edit-Distanzen zwischen Wörtern zur Anwendung. Darüber hinaus wurden zur weiteren Verfeinerung des linguistischen Matching-Ansatzes das in der Microsoft Windows-Systemumgebung verfügbare Kommandozeilentool *TreeTagger*[23] zur Auszeichnung von Wortklassen in einem deutschen Satz oder Satzteil und *jWordSplitter*[24] zur Zerlegung deutscher Komposita verwendet. Der *TreeTagger* basiert auf dem probabilistischen Markov-Model und erweitert dieses für die deutsche Sprache [Schmid 1995; 1994]. Schließlich kam ein anwendungsspezifischer, Synonyme verzeichnender Thesaurus, zu dessen Erstellung der deutschsprachige Open-Source-Thesaurus *openthesaurus.de* [25] herangezogen wurde, zur Anwendung. Die Version der verwendeten ConedaKor-Datenbank war Version 1.8.3.

5.1 Laden der zu mappenden Datenmodelle in ein gemeinsames objektorientiertes Datenmodell

5.1.1 CIDOC CRM

Die bereits im objektorientierten Datenmodell realisierte CIDOC CRM-Datenstruktur in RDF-XML-serialisierter Form wird aus einem in UTF-8 kodierten Textdokument in eine Struktur aus Programmobjekten geladen. Als Implementierung des CIDOC CRM

[21] https://github.com/ruby-rdf/rdf [30.03.2015].
[22] https://github.com/GlobalNamesArchitecture/damerau-levenshtein [30.03.2015].
[23] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ [01.05.2015].
[24] http://www.danielnaber.de/jwordsplitter/ [01.05.2015].
[25] https://www.openthesaurus.de/ [01.05.2015].

wurde das an der Friedrich-Alexander-Universität in Nürnberg-Erlangen in Zusammenarbeit mit dem Germanischen Nationalmuseum in Nürnberg und dem Zoologischen Forschungsmuseum Alexander König in Bonn entwickelte Erlangen CRM / OWL 2012-01-11[26] verwendet, das auf der letzten offiziellen Version 5.0.4 des CIDOC CRM beruht. Zusätzlich wurden die englischen Namen der implementierten CRM-Ressourcen im Textdokument durch ihre deutschen Referenzübersetzungen[27] ersetzt.

Mittels der oben genannten Ruby-Gems *rdf* und *rdf-xml* werden die in RDF-XML-serialisierten und mittels der Ontologiesprachen RDFS und OWL definierten Elemente des CIDOC CRM, d.h. Klassen und Properties, aus dem Textdokument ausgelesen und als Programmobjekte instanziiert.

Dies geschieht zunächst für die Klassen des CRM, die in einem ersten Schritt unter anderem als eindeutigen Bezeichner ihre URI und als natürlichsprachlichen Namen die deutsche Version des Ressourcennamens zugewiesen bekommen. In einem zweiten Schritt wird die Hierarchie des CRMs zwischen Klassen hergestellt und für jedes eine CRM-Klasse repräsentierende Objekt die direkten Ober- und Unterklassen gesetzt.

Für die Properties des CRM wird in einem ersten Schritt genauso vorgegangen, zusätzlich werden die Verweise auf Domain- und Range-Klasse der Property gesetzt. In einem zweiten Schritt werden auch hier Ober- und Unterproperties sowie, sofern existent, die inverse Property gesetzt. Da für die verwendete Version des Erlangen CRMs nicht für alle Properties Domain-und Range-Klasse direkt definiert wurde, wurde zusätzlich ein auf den durch RDFS/OWL definierten logischen Abhängigkeiten zwischen den Ressourcen operierender, einfacher Inferenzmechanismus implementiert, der Domain- und Range-Klassen einer Property, sofern nicht explizit angegeben, von der inversen Property oder den Oberproperties der betrachteten Property ableitet.

5.1.2 ConedaKor

Im Gegensatz zum CIDOC CRM ist das Datenmodell von ConedaKor nicht objektorientiert, sondern graphbasiert (siehe Kapitel 2.1) und muss erst in ein gemeinsames objektorientiertes Datenmodell überführt werden. Die zu mappenden Schemadaten (siehe Kapitel 2.1.2) von ConedaKor werden mittels des ActiveRecord-

[26] http://erlangen-crm.org/120111/ [30.03.2015].
[27] http://cidoc-crm.gnm.de/wiki/Hauptseite [30.05.2015].

Moduls von Rails aus der ConedaKor zugrunde liegenden, relationalen MySQL-Datenbank ausgelesen und als Objekte instanziiert.

Dies geschieht zunächst für Kor-Entitätstypen (*Kind*) und für Kor-Relationen (*Relation*), wobei wegen der oben angesprochenen freien Modifizierbarkeit der Definitionen der Relationen Verknüpfungen zwischen Entitäten existieren können, die durch die aktuelle Definition nicht abgedeckt werden. Damit diese Verbindungen dennoch für das Mapping erhalten bleiben, werden alle „eigentlichen" Relationen (*ActualRelations*) aus den Instanzdaten, d.h. den konkreten Verknüpfungen zwischen Entitäten des ConedaKor-Graphen, abgeleitet und mit der Relation assoziiert.

In einem zweiten Schritt werden die Felder der Kor-Entitätstypen ebenfalls in Entitätstypen und Relationen überführt, um ein einheitliches Datenmodell herzustellen. Dabei werden für die statischen Felder *Name* und *eindeutiger Name*, die jeder Entitätstyp hat, sowie für dynamische Felder[28], die sowohl für Entitätstypen als auch für einzelne Entitäten angelegt werden können, jeweils ein neuer Entitätstyp mit dem Namen des Feldes sowie eine Relation mit dem Namen bzw. inversen Namen „hat [Name des Feldes]" bzw. „ist [Name des Feldes] von" als Programmobjekte angelegt. Damit die Anzahl der zu mappenden Elemente überschaubar bleibt, werden die so transformierten dynamischen Felder auf solche beschränkt, die von einer Mindestanzahl von Entitäten verwendet werden.

Die zu etablierenden semantischen Korrespondenzen zwischen den Ressourcen von ConedaKor und CIDOC CRM werden durch Referenzen von Entitätstypen auf ihre korrespondierende Klasse im CRM sowie durch die Referenzen von „eigentlichen" Kor-Relationen auf Properties und zwischen diesen artikulierenden Klassen, die einen der Kor-Relation semantisch entsprechenden Property-Pfad bilden, realisiert.

5.2 Alignment-Serialisierung und -Deserialisierung

Die Kommunikation zwischen Anwender und in Rails realisierter Webanwendung beruht auf dem zustandslosen HTTP-Protokoll. Zum Vermeiden des ständigen Nachladens der zu mappenden Datenstrukturen aus ihren ursprünglichen Quellen sowie zur Persistierung bereits etablierter Mappings werden die die Ressourcen aus ConedaKor und dem CRM repräsentierenden Programmobjekte in JSON (JavaScript

[28] In der aktuellen Version nur Felder zur Datierung.

Object Notation) in Textdokumenten serialisiert bzw. von diesen aus wieder als Programmobjekte deserialisiert.

Zum Serialisieren überschreiben die die zu mappenden Elemente beschreibenden Klassen die Methode *as_json*, die die Struktur des zu serialisierenden Objekts in einen sogenannten Hash, d.h. ein assoziatives Datenfeld, überführt. Damit es dabei nicht zu einem bedeutenden Aufwand durch die automatische Serialisierung weiterer vom Objekt aus referenzierter Objekte und sogar zu Zirkelreferenzen zwischen diesen kommt, werden die Referenzen auf andere Objekte durch die eindeutigen Bezeichner der durch diese Objekte repräsentierten Kor- bzw. CRM-Elemente ersetzt. Im Falle von Elementen des CRMs sind dies URIs, im Falle von Elementen aus ConedaKor IDs.

Durch Aufruf der Methode *to_json* auf den zu serialisierenden Objekten, werden diese zunächst durch die Methode *as_json* in Hashs überführt bevor die Hashs dann im JSON-Format in ein UTF-8-kodiertes Textdokument serialisiert werden.

Zum Deserialisieren implementieren alle die zu mappenden Elemente beschreibenden Klassen die Klassenmethode *json_create*, der als Parameter die das zu deserialisierende Objekt repräsentierende Zeichenkette übergeben wird.

Diese Methode instanziiert zunächst das Objekt, ohne allerdings die Referenzen auf andere Objekte aufzulösen, da diese unter Umständen noch gar nicht instanziiert wurden. Zur Auflösung der Objektreferenzen definiert jede Klasse die Instanzmethode *reestablishLinks*, der als Parameter die bereits instanziierten Objekte, auf die die Objektart verweist, übergeben werden. Mittels der deserialisierten eindeutigen Objektbezeichner, d.h. URIs bzw. IDs, werden dann die Objektreferenzen auf die entsprechenden Objekte wieder hergestellt.

5.3 Architektur des Mapping-Tools – Model-View-Controller-Architektur

5.3.1 Anwendungsfälle

Der *Kor-CRM-Mapper* deckt mehrere Anwendungsfälle ab (Abbildung 19). Zunächst einmal erlaubt er dem Anwender sich die zu mappenden Kor-Entitätstypen und Kor-Relationen anzeigen zu lassen. Von der jeweiligen Listendarstellung aus sind sowohl das Anlegen von Mappings sowie das Entfernen bereits erstellter Mappings möglich.

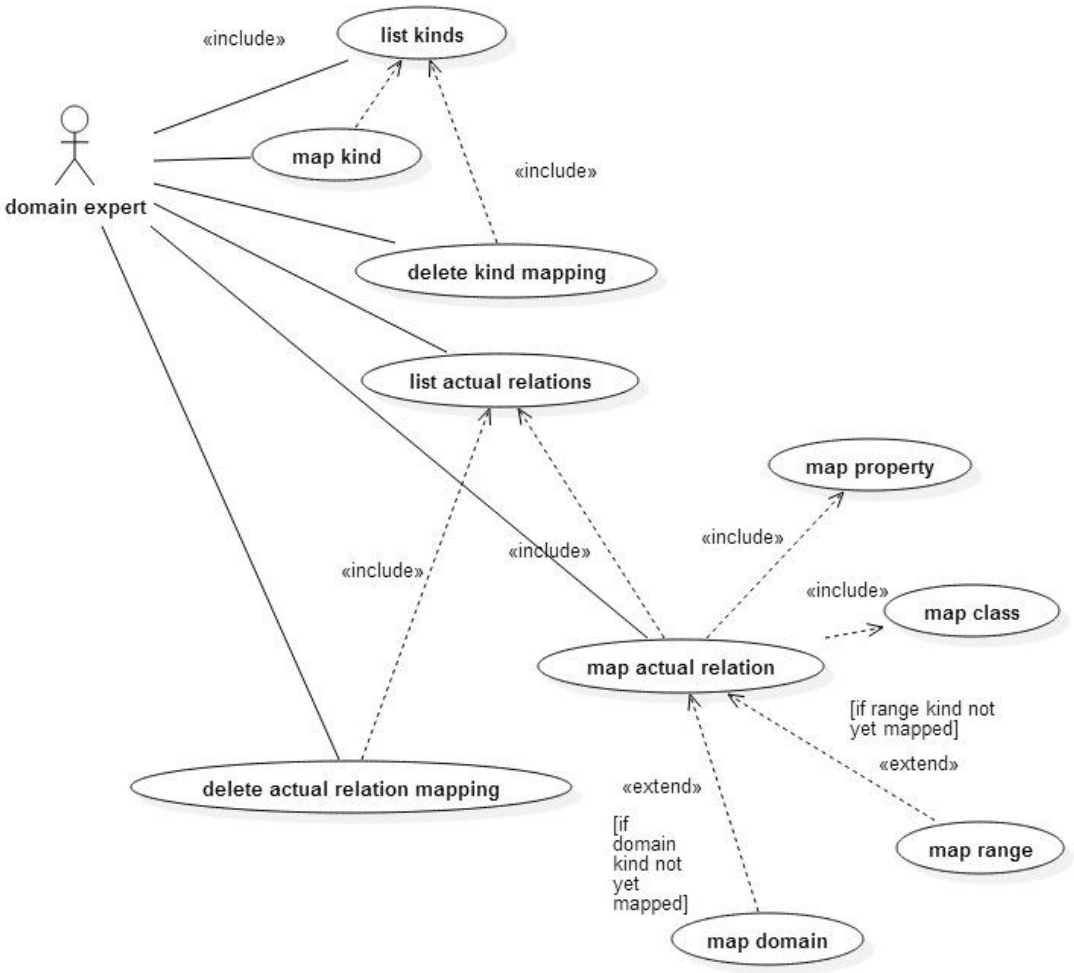

ABBILDUNG 19. VON KCM REALISIERTE ANWENDUNGSFÄLLE

Das Mapping von Kor-Relationen setzt sich iterativ solange aus der Erstellung eines der Relation entsprechenden Property-Pfades aus Properties und intermediären Klassen des CRMs zusammen bis ein Property-Pfad konstituiert worden ist, der die mit der Domain und Range der Relation gemappten CRM-Klassen miteinander verbindet.

Im Falle, dass Domain und/oder Range der Kor-Relation noch nicht mit Klassen des CRMs gemappt worden sind, geschieht dies in einem ersten Schritt bevor der Property-Pfad konstituiert werden kann.

5.3.2 Model-Klassen

Abbildung 20 stellt die Klassen der zu mappenden CRM-Ressourcen dar. *CrmClass* und *CrmProperty* sind beide Unterklassen der Klasse *CrmRessource*, die gemeinsame Eigenschaften enthält wie die URI als eindeutigen Bezeichner der Ressourcen (*uri*),

deren (deutschen) natürlichsprachlichen Namen (*label*), den im CRM vorgesehenen Code (*notation*), den aus dem Code abgeleiteten numerischen Wert (*number*), die die Semantik der Ressource beschreibende *Scope Note* (*comment*), sowie einen Ähnlichkeitswert (*similarity*), der für das automatische Matching mit Elementen aus ConedaKor jeweils für jeden Fall neu berechnet wird (siehe Kapitel 5.4).

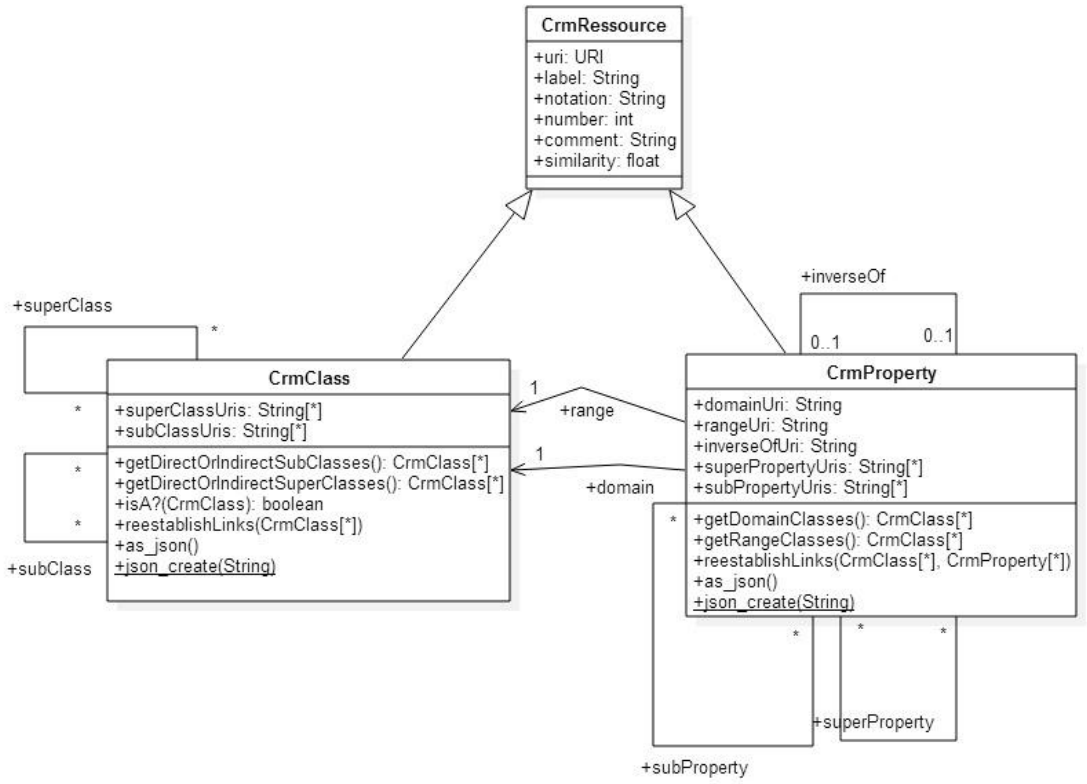

ABBILDUNG 20. CRM-RESSOURCEN REPRÄSENTIERENDE KLASSEN

Objekte der Klasse *CrmClass* können eine beliebige Anzahl von Ober-und Unterklassen haben. Die Attribute *superClassUris*, *subClassUris* sowie die Methoden *reestablishLinks*, as*_json* und die statische Methode json*_create* dienen der oben beschriebenen Serialisierung bzw. Deserialisierung der Objekte.

Die Methode *getDirectOrIndirectSubClasses* sowie *getDirectOrIndirectSuperClasses*, liefern als Rückgabewert alle direkten und indirekten Subklassen im ersten bzw. alle direkten und indirekten Superklassen im zweiten Fall zurück.

Die Methode *isA?* testet, ob es sich bei dem Objekt um eine Subklasse des als Parameter übergebenen Objekts handelt, in welchem Falle die Methode den Wert *true*, sonst *false* zurückliefert.

Objekte der Klasse *CrmProperty* haben, falls vorhanden, Referenzen auf ihre inverse Property (*inverseOf*), beliebig viele Ober- und Unterproperties (*subProperties*, *subProperties*) sowie auf jeweils genau eine Domain- und Range Klasse (*domain*, *range*).

Wie bei Objekten der Klasse *CrmClass* dienen die Attribute *domainUri*, *rangeUri*, *inverseOfUri*, *superPropertyUris*, *subPropertyUris* sowie die Methoden *reestablishLinks*, *as_json* und *json_create* der Serialisierung bzw. Deserialisierung der Objekte der Klasse.

Die Methoden *getDomainClasses* bzw. *getRangeClasses* liefern alle Objekte der Klasse *CrmClass* zurück, die als Domain bzw. Range der Property in Frage kommen.

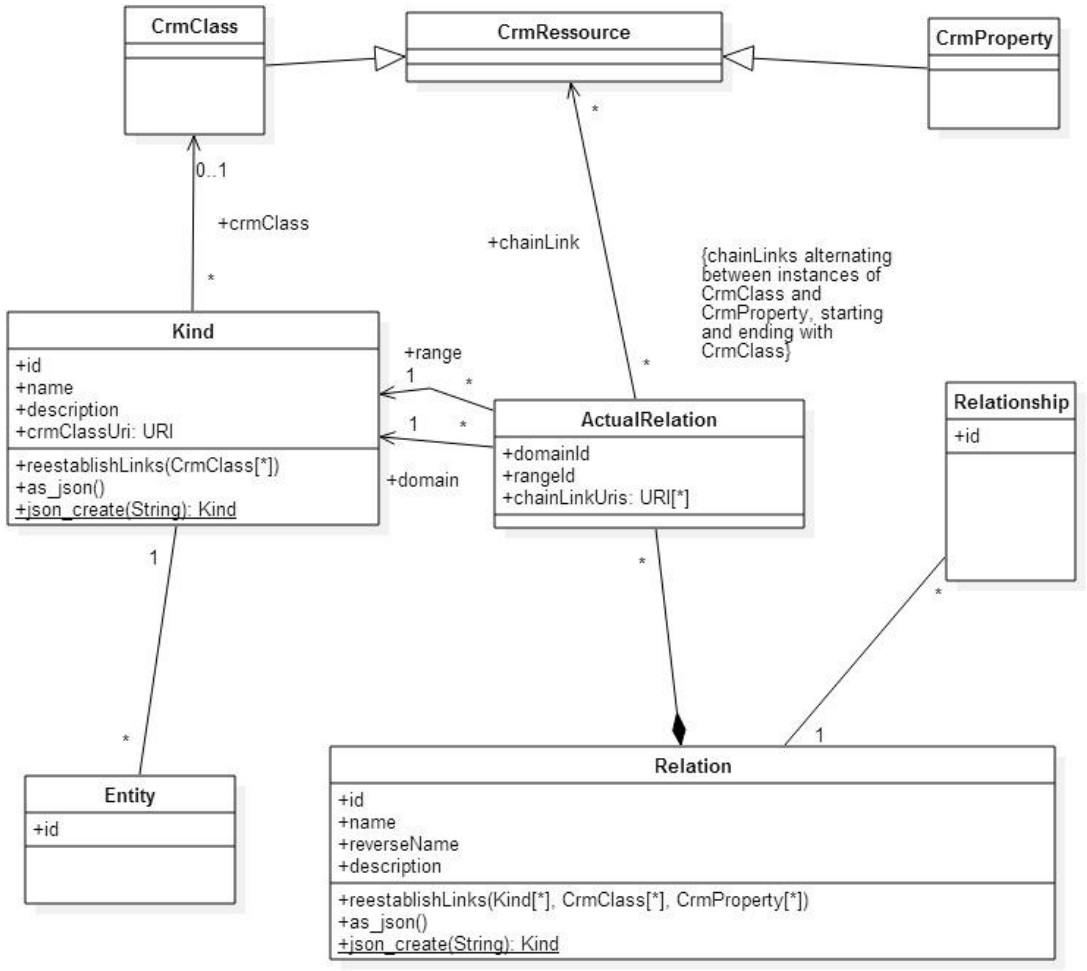

ABBILDUNG 21. KOR-ELEMENTE REPRÄSENTIERENDE KLASSEN

Abbildung 21 stellt die die Kor-Elemente repräsentierenden Model-Klassen dar. Entitätstypen klassifizieren Entitäten (*Entity*), Relationen klassifizieren

Verknüpfungen (*Relationship*). Relationen setzten sich wie bereits oben dargestellt aus „eigentlichen“ Relationen (*ActualRelationship*) zusammen, die die verschiedenen Kombinationen von Domain und Range für Entitätstypen, die in den Instanzdaten tatsächlich vorkommen, darstellen.

Die Klasse *Kind* wird durch ihren eindeutigen Bezeichner (*id*), ihren Namen (*name*) und ihre Beschreibung (*description*) beschrieben. Die Korrespondenz zu einer Klasse des CRM wird durch die Objektreferenz *crmClass* ausgedrückt.

Das Attribut *crmClassUri*, sowie die Methoden *reestablishLinks*, *as_json* und *json_create* dienen wieder der Serialisierung bzw. Deserialisierung der Objekte der Klasse.

Objekte der Klasse *Relation* haben wie Entitätstypen die Attribute *id*, *name* und *description*. Da es sich bei den Kor-Relationen grundsätzlich um bidirektionale Verbindungen zwischen Entitätstypen handelt, definiert das Attribut *reverseName* die Semantik der umgekehrten Verbindung von Range- zu Domain-Entitätstyp.

Die „eigentlichen“ Relationen der Relation definieren darüber hinaus die konkreten Entitätstypen, zwischen denen eine entsprechende Relation bestehen kann (*range*, *domain*). Über eine geordnete Liste von *chainLink*-Objektreferenzen auf CRM-Ressourcen wird die Korrespondenz zu einem Pfad von CRM-Properties hergestellt. Dabei startet der Pfad bei der mit dem Domain-Entitätstyp der Relation gemappten CRM-Klasse und endet mit der mit dem Range-Entitätstyp der Relation gemappten CRM-Klasse. Zwischen Start- und Endklasse alternieren CRM-Properties und CRM-Klassen.

domainId, *rangeId* und *chainLinkUris* sind die Attribute der „eigentlichen“ Relationen die mittels der Methoden *reestablishLinks*, *as_json* und *json_create* der Serialisierung bzw. Deserialisierung der Relationen dienen.

5.3.3 Controllers und Views

Am Anfang jeder Controller-Aktion steht das erneute Deserialisieren der zu mappenden Objekte der Klassen *CrmClass*, *CrmProperty*, *Kind* und *Relation*, da die Programmobjekte in der zustandslos ausgeführten Webanwendung nicht zwischen mehreren Anfragen erhalten bleiben.

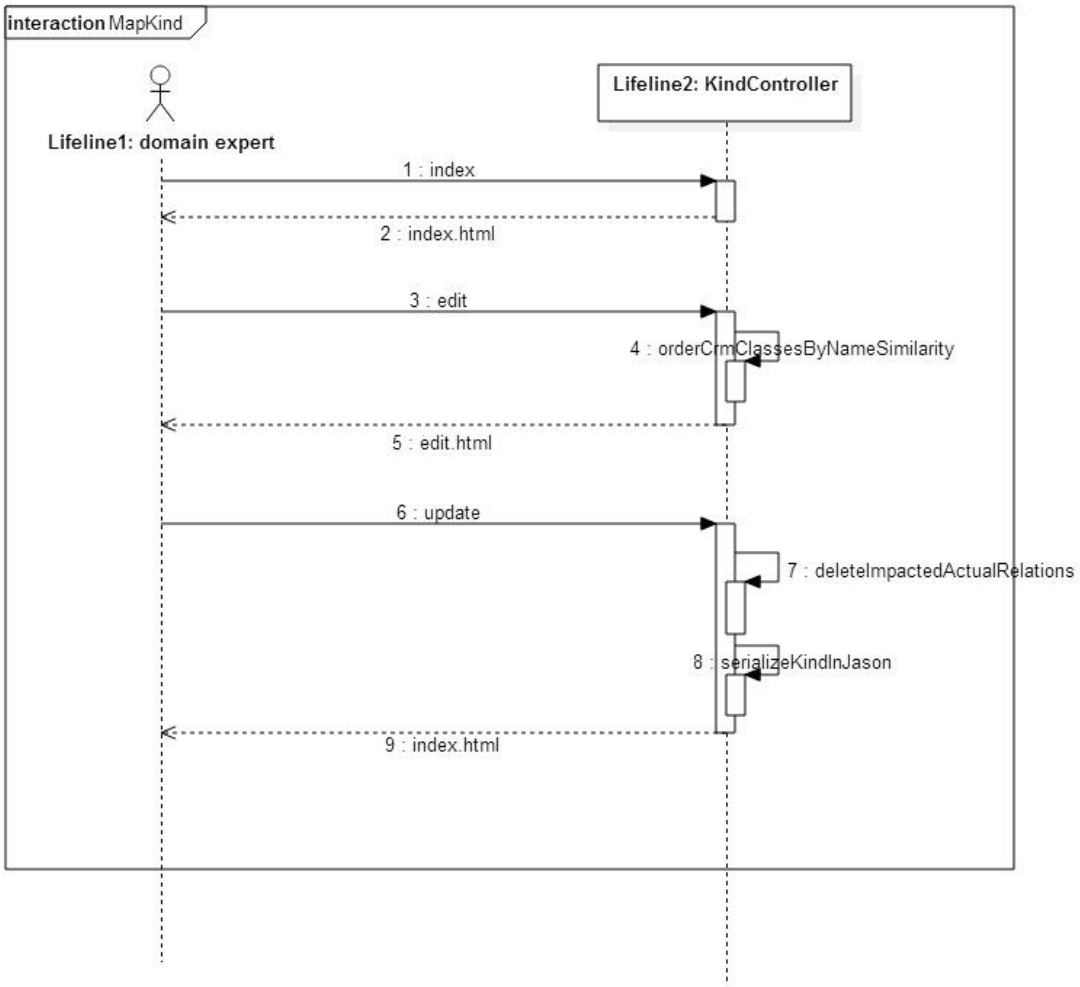

ABBILDUNG 22. INTERAKTIONSDIAGRAMM DES MAPPENS VON ENTITÄTSTYPEN

Zum Mappen von Entitätstypen (Abbildung 22) wird zunächst einmal durch Aufruf der Aktion *index* eines Objekts der Klasse *KindController* dem Anwender eine Liste der zu mappenden Entitätstypen[29] dargestellt (Abbildung 23).

Bei Klick auf den zu einem Entitätstyp gehörenden Link „Edit“, wird die Aktion *edit* des Controllers aufgerufen. Diese ermittelt aufgrund des über die Anfrage übergebenen *id*-Parameters den gewünschten Entitätstyp und führt einen terminologisch-linguistischen Ähnlichkeitsvergleich zwischen dem Namen des Entitätstyps und den Namen aller CRM-Klassen aus, die auf den ermittelten Ähnlichkeitswerten basierend in der daraufhin dem Anwender anzuzeigenden Auswahlliste nach absteigendem Ähnlichkeitswert geordnet werden. Die Berechnung

[29] Und Felder (siehe Kapitel 5.1.2).

der lexikalischen Ähnlichkeitswerte der Namen wird in Kapitel 5.4.1 näher beschrieben.

Entitätstypen

Entitaetstyp	Entsprechende CRM-Klasse	
Medium		Edit Destroy
Ort		Edit Destroy
Institution		Edit Destroy
Person	Person	Edit Destroy
Literatur	Informationsträger	Edit Destroy
Werk		Edit Destroy
Text		Edit Destroy
Personengruppe		Edit Destroy
Ausstellung		Edit Destroy
Embedded Medium		Edit Destroy
Name		Edit Destroy
Eindeutiger Name		Edit Destroy
Aktiv		Edit Destroy
Baubeginn		Edit Destroy
Bauzeit		Edit Destroy
Datierung		Edit Destroy

ABBILDUNG 23. LISTENDARSTELLUNG DER KOR-ENTITÄTSTYPEN SOWIE DER ALS ENTITÄTSTYPEN ABGELEITETEN FELDER

Der Anwender kann im zurückgelieferten Formular (Abbildung 24) den ausgewählten Kor-Entitätstyp mit der in der Auswahlliste an oberster Stelle stehenden CRM-Klasse durch Klick auf den Button „Save Kind“ direkt mappen oder in der geordneten Auswahlliste nach einer semantisch geeigneteren CRM-Klasse suchen und diese auswählen. Durch einen Klick auf den Link „Back“ kann der Vorgang auch abgebrochen werden und der Anwender gelangt wieder zur Listendarstellung der Entitätstypen zurück.

Wird der Button „Save Kind“ angeklickt, wird die Aktion *update* des Controllers aufgerufen. Diese ermittelt zunächst wieder das Objekt des betroffenen Entitätstyps sowie das die ausgewählte CRM-Klasse repräsentierende Objekt mittels der mit der Anfrage übermittelten Parameter. Handelt es sich beim Mapping nicht um ein initiales Mapping zwischen einem Entitätstyp und einer CRM-Klasse und bestehen

bereits Mappings zwischen „eigentlichen“ Relationen und CRM-Property-Pfaden, deren Domain oder Range der neu zu mappende Entitätstyp ist, wird überprüft, ob es sich bei der neu ausgewählten CRM-Klasse nicht um eine Unterklasse der bereits gemappten Klasse handelt und wenn dies nicht der Fall ist, die betroffenen „eigentlichen“ Relationen gelöscht. Danach erst wird dem Entitätstyp-Objekt die neue CRM-Klasse zugewiesen und das Objekt serialisiert. Anschließend wird zur Anzeige der Listendarstellung zur Aktion *index* umgeleitet.

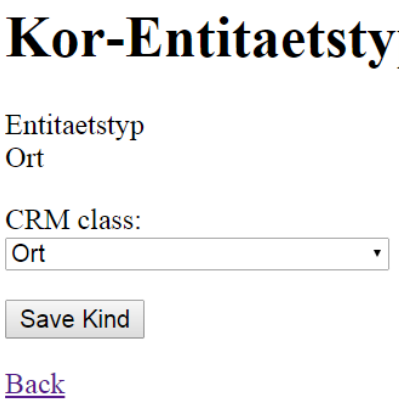

ABBILDUNG 24. FORMULAR ZUM MAPPEN EINES KOR-ENTITÄTSTYP MIT EINER KLASSE DES CRMS

In der Listendarstellung der Entitätstypen steht dem Anwender auch die Möglichkeit des Löschens bereits erstellter Mappings zur Verfügung. Durch Klick auf den Link „Destroy“ wird die Aktion *destroy* des Controllers aufgerufen, der wieder über die übermittelten Parameter zunächst den betroffenen Entitätstyp ermittelt. Dann werden die eventuell durch die Auflösung des Mappings betroffenen „eigentlichen„ Relationen ermittelt und gelöscht. Schließlich wird die Referenz auf die gemappte CRM-Klasse entfernt und das Entitätstyp-Objekt serialisiert. Anschließend wird wieder zur *index*-Aktion umgeleitet.

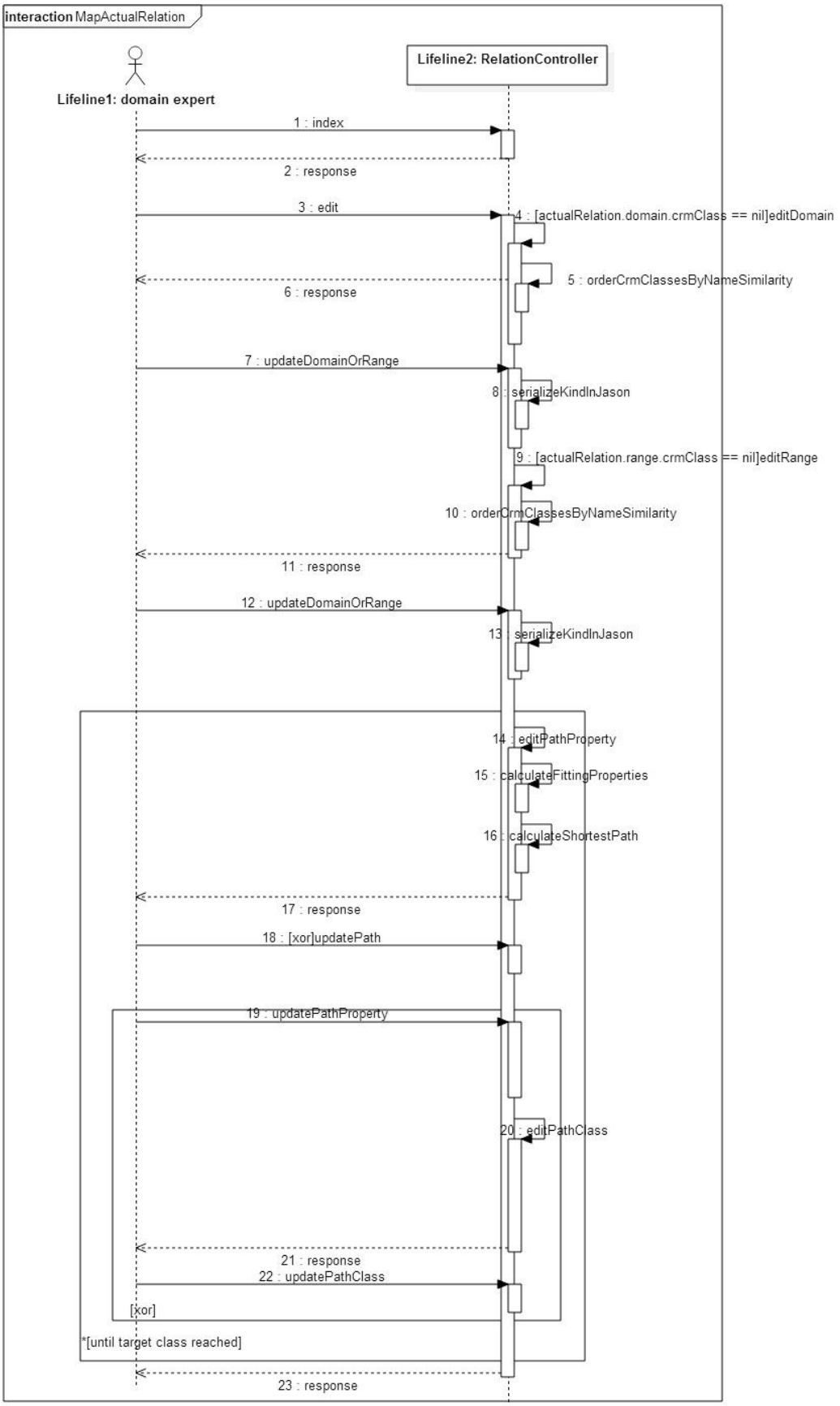

ABBILDUNG 25. INTERAKTIONSDIAGRAMM DES MAPPENS VON „EIGENTLICHEN“ RELATIONEN

Zum Mappen der Kor-Relationen (Abbildung 25) wird ähnlich wie bei den Entitätstypen zunächst nach Anzeige der Liste der zu mappenden „eigentlichen“ Relationen vom Anwender durch Klick auf den zur zu mappenden „eigentlichen“ Relation gehörigen Link „Edit“ die Aktion *edit* des *RelationControllers* aufgerufen. Nachdem das entsprechende, die „eigentliche“ Relation repräsentierende Objekt mittels der übertragenen Parameter gefunden worden ist, wird überprüft, ob Domain- und Range-Entitätstypen der Relation bereits mit CRM-Klassen gemappt worden sind. Ist dies nicht der Fall wird zu den Aktionen *editDomain* bzw. *editRange* umgeleitet. Diese verfahren analog zur bereits oben beschriebenen *edit*-Aktion der Klasse *KindController*. Nachdem vom Anwender die dem Entitätstyp entsprechende CRM-Klasse ausgewählt worden ist, wird durch Klick auf „OK“ die *updateDomainOrRange*-Aktion der Klasse *RelationController* ausgeführt. Auch diese Aktion verfährt analog zur *update*-Aktion der Klasse *KindController*, mit dem Unterschied, dass hier nicht auf eventuell vom Entitätstyp-Mapping betroffene „eigentliche“ Relationen überprüft werden muss. Nach Serialisierung des Entitätstyp-Mappings wird wieder zur *edit*-Aktion umgeleitet. Sind sowohl Domain- als auch Range-Entitätstyp gemappt, wird für das Mapping eine Liste instanziiert, und mit einer Session, d.h. einer stehenden Verbindung zwischen Client und Server, assoziiert, die für die Durchführung der laufenden Mapping-Aktion erhalten bleibt. Das erste Element der Liste verweist auf die mit dem Domain-Entitätstyp gemappte CRM-Klasse, das letzte Element auf die mit dem Range-Entitätstyp gemappte CRM-Klasse. Danach wird zur Aktion *editPathProperty* weitergeleitet.

Diese Aktion berechnet alle Properties des CRMs, die als nächstes Glied im zu konstituierenden Property-Pfad von der letzten Klasse im Pfad aus gültig sind, d.h. diejenigen Properties, deren Domain-Klasse eine Oberklasse der letzten Klasse im Pfad ist.

Zusätzlich wird ein kompletter „kürzester“ Pfad aus Properties und intermediären Klassen von der letzten Klasse im Pfad aus zur mit dem Range-Entitätstyp der „eigentlichen“ Relation gemappten CRM-Klasse berechnet. Diese Berechnung wird in Kapitel 5.4.2 beschrieben.

Im dem Anwender zurückgelieferten Formular (Abbildung 26) kann vom Anwender nun eine der zuvor berechneten Properties ausgewählt werden, die den nächsten Schritt im Property-Pfad von Domain zu Range darstellt. Durch Klick auf den unter der Auswahlliste stehenden Button „OK“ wird die Aktion *updatePathProperty* der Klasse *RelationController* aufgerufen. Entspricht der berechnete komplette „kürzeste“ Pfad von der aktuell letzten Klasse im bisher konstituierten Property-Pfad zur Range semantisch der zu mappenden „eigentlichen“ Relation, kann durch Klick auf den Button „OK“ hinter der Angabe des Pfades der ganze Pfad gleich übernommen

werden. In diesem Fall wird die Aktion *updatePath* des Controllers aufgerufen. Durch Klick auf den Link „Abbruch“ wird der Mapping-Vorgang für die ausgewählte „eigentliche“ Relation abgebrochen und wieder auf die Listendarstellung umgeleitet.

ABBILDUNG 26. *EDITPATHPROPERTY*-FORMULAR

Auch in der Aktion *updatePath* werden zunächst einmal die relevanten Objekte wieder aufgefunden. Die mit der Session assoziierte Liste wird um die Properties und Klassen des CRM, die den vorgeschlagenen „kürzesten“ Pfad konstituieren, ergänzt und dieser dem „eigentlichen“ Relation-Objekt zugewiesen, welches, da die mit der Range gemappte CRM-Klasse oder eine ihrer Unterklassen durch den Pfad erreicht worden ist, serialisiert werden kann. Danach wird wieder auf die Listendarstellung umgeleitet.

Wurde nicht der komplette Pfad gemappt, sondern nur die nächste Property auf dem Pfad, fügt die Aktion *updatePathProperty*, nachdem die relevanten Objekte aufgefunden worden sind, die ausgewählte Property der mit der Session assoziierten, den Pfad repräsentierenden Liste hinzu. Ist die Range der so hinzugefügten Property die mit der Range der „eigentlichen“ Relation gemappten CRM-Klasse oder eine ihrer Oberklassen, ist das Ziel erreicht und die Session-Liste wird dem „eigentlichen“ Relation-Objekt zugewiesen, welches daraufhin serialisiert wird. Anschließend wird zur Listendarstellung weitergeleitet.

Ist das Ziel nicht erreicht, so wird im nächsten Schritt die spezifischste Klasse ausgewählt, die auf dem Property-Pfad der Range der gerade dem Pfad hinzugefügten Property entspricht. Damit stehen im nächsten Schritt der Auswahl einer weiteren Property des Pfades spezifischere Properties zur Verfügung, die von

dieser Unterklasse ausgehen können. Zu diesem Zwecke wird zur Aktion *editPathClass* umgeleitet.

Die Aktion *editPathClass* berechnet alle passenden CRM-Klassen, d.h. die Klasse, die der Range der letzten CRM-Property des Property-Pfades entspricht sowie alle ihre Unterklassen. Die berechneten passenden CRM-Klassen werden dem Anwender in einer Auswahlliste in einem Formular zurückgeliefert. Nachdem der Anwender die gewünschte Klasse ausgewählt hat, wird durch Klick auf den Button „OK“ die Aktion *updatePathClass* aufgerufen. Ein Klick auf den Link „Abbruch“ beendet den Mapping-Vorgang für die ausgewählte „eigentliche“ Relation und es wird auf die Listendarstellung der Relationen umgeleitet.

updatePathClass fügt die ausgewählte Klasse der mit der Session assoziierten, den Property-Pfad repräsentierenden Liste hinzu und leitet zur Aktion *editPathProperty* weiter. Das Alternieren zwischen e*ditPathProperty*/*updatePathProperty* und *editPathClass*/*updatePathClass* findet solange statt, bis der Vorgang durch Klick auf den Link „Abbruch“ abgebrochen wird oder durch die letzte dem Property-Pfad hinzugefügte Property die mit der Range der „eigentlichen“ Relation gemappte CRM-Klasse oder eine ihrer Subklassen erreicht wird.

Die Aktion *destroy* schließlich, die in der Listendarstellung durch Klick auf den mit einer „eigentlichen“ Relation assoziierten Link „Destroy“ aufgerufen wird, reinitialisiert das Mapping der „eigentlichen“ Relation.

5.4 Matching

5.4.1 Matching von Kor-Entitätstypen und CRM-Klassen

Beim Mapping von Kor-Entitätstypen und CRM-Klassen wird davon ausgegangen, dass je ähnlicher sich die Namen eines Entitätstyps und einer Klasse sind, es auch desto wahrscheinlicher ist, dass beide dasselbe semantische Konzept repräsentieren.

Aus diesem Grunde wurde eine Methode implementiert, die die Namen des zu mappenden Kor-Entitätstyps paarweise mit allen Namen der CRM-Klassen vergleicht und diesen auf Grundlage des terminologisch-linguistischen Vergleichs Ähnlichkeitswerte zuordnet. Diese Werte werden genutzt, um die am besten passende CRM-Klasse zu ermitteln bzw. alle CRM-Klassen absteigend nach der Ähnlichkeit ihres Namens mit dem Namen des Entitätstyps in einer Liste zu ordnen.

Die terminologische Ähnlichkeit zwischen Bezeichnern von Kor-Entitätstypen und CRM-Klassen wird durch einen Edit-Distanz-Wert, für den die Levenshtein-Metrik verwendet wird, bestimmt. Zur Verbesserung des Vergleichs wurden zusätzliche linguistische Ansätze implementiert und für jedes zu vergleichende Element, d.h. Kor-Entitätstypen und CRM-Klassen angewendet (Tabelle 2).

Def orderCrmClassesByNameSimilarity (kind, crmClasses)		
kindNounPhraseConstituents = KorCrmMatching::NounPhraseAnalyser.analyseNounPhrase kind.name		
decomposeHeads kindNounPhraseConstituents		
kindNounPhraseHeads = getHeads kindNounPhraseConstituents		
kindNounPhraseModifiers = getModifiers kindNounPhraseConstituents		
kindNameSynonyms = KorCrmMatching::SynonymsProvider.getSynonyms kindNounPhraseHeads		
For each crmClass **in** crmClasses		
crmClassNounPhraseConstituents = crmClass.nounPhraseConstituents		
crmClassNounPhraseHeads = getHeads crmClassNounPhraseConstituents		
crmClassNounPhraseModifiers = getModifiers crmClassNounPhraseConstituents		
crmClassNameSynonyms = KorCrmMatching::SynonymsProvider.getSynonyms crmClassNounPhraseHeads		
crmClass.headSimilarity = calculateBestSimilarity kindNameSynonyms, crmClassNameSynonyms		
If kindNounPhraseModifiers.empty? && crmClassNounPhraseModifiers.empty?		
True		**False**
crmClass.modifierSimilarity = 1		crmClass.modifierSimilarity = calculateBestSimilarity kindNounPhraseModifiers, crmClassNounPhraseModifiers
orderedClasses = orderClassesByHeadAndModifierSimilarity crmClasses		
Return orderedClasses		

TABELLE 2. STRUKTOGRAMM DES AUF TERMINOLOGISCHE ÄHNLICHKEIT BERUHENDEN ORDNENS DER KLASSEN DES CRM IN BEZUG AUF EINEN ENTITÄTSTYP AUS CONEDAKOR

In einem ersten Schritt werden die Nominalphrasen der Namen der zu mappenden Elemente analysiert[30]. Dabei wird von einer vereinfachten Nominalphrasenstruktur ausgegangen, die die Komplexität allgemeiner Nominalphrasen einschränkt und der dabei dennoch alle Nominalphrasen der Bezeichner der CRM-Klassen sowie der

[30] Zur grammatischen Struktur deutscher Sätze und insbesondere Nominalphrasen: Grammis 2.0. das grammatische informationssystem des instituts für deutsche sprache (ids): http://hypermedia.ids-mannheim.de/index.html [09.05.2015].

aktuell in der Frankfurter ConedaKor-Installation existierenden Kor-Entitätstypen entsprechen[31].

Def analyseNounPhrase (nounPhrase)
cleanedNounPhrase = clean nounPhrase
taggedNounPhrase = tagNounPhrase cleanedNounPhrase
nounPhraseConstituents = decomposeNounPhrase taggedNounPhrase
normalizeCompounds! nounPhraseConstituents
Return nounPhraseConstituents

TABELLE 3. STRUKTOGRAMM DER NOMINALPHRASENANALYSE

Bei der Analyse (Tabelle 3) wird in einem ersten Schritt die den Bezeichner darstellende Nominalphrase „gereinigt“ und dadurch normalisiert. Die aktuell durchgeführte Operation sieht das Entfernen von Klammerausdrücken vor. In einem nächsten Schritt werden die Wörter der Nominalphrase mittels des POS-Taggers, der den einzelnen Wörtern Wortklassen zuordnet und die Wörter durch Lemmatisierung, d.h. der heuristischen Anwendung von Regeln, auf ihre Grundform zurückführt, ausgezeichnet. In einem nächsten Schritt wird die so getaggte Nominalphrase zerlegt.

Da das CRM komposite Konzepte, d.h. Konzepte die sich explizit aus mehreren Konzepten zusammensetzen, beinhaltet, gibt es Bezeichner, deren Nominalphrasen sich wiederrum aus mehreren durch beiordnende Konjunktionen verbundene Nominalphrasen zusammensetzen, so beispielsweise *E29 Design or Procedure* (*E29 Entwurf oder Verfahren*). Diese Nominalphrasen werden in einem ersten Schritt in ihre durch die beiordnenden Konjunktionen verbundenen Nominalphrasen-komponenten zerlegt, bevor diese weiter analysiert werden (Tabelle 4).

[31] Damit der Algorithmus auch in Zukunft anwendbar bleibt, könnte für die zukünftige Benennung von Kor-Entitätsklassen Guidelines entwickelt werden, die die Nutzer auf die Eingabe von Bezeichnern von Entitätstypen gemäß der im CRM bisher verwendeten Nominalphrasenstruktur der Bezeichner einschränkt.

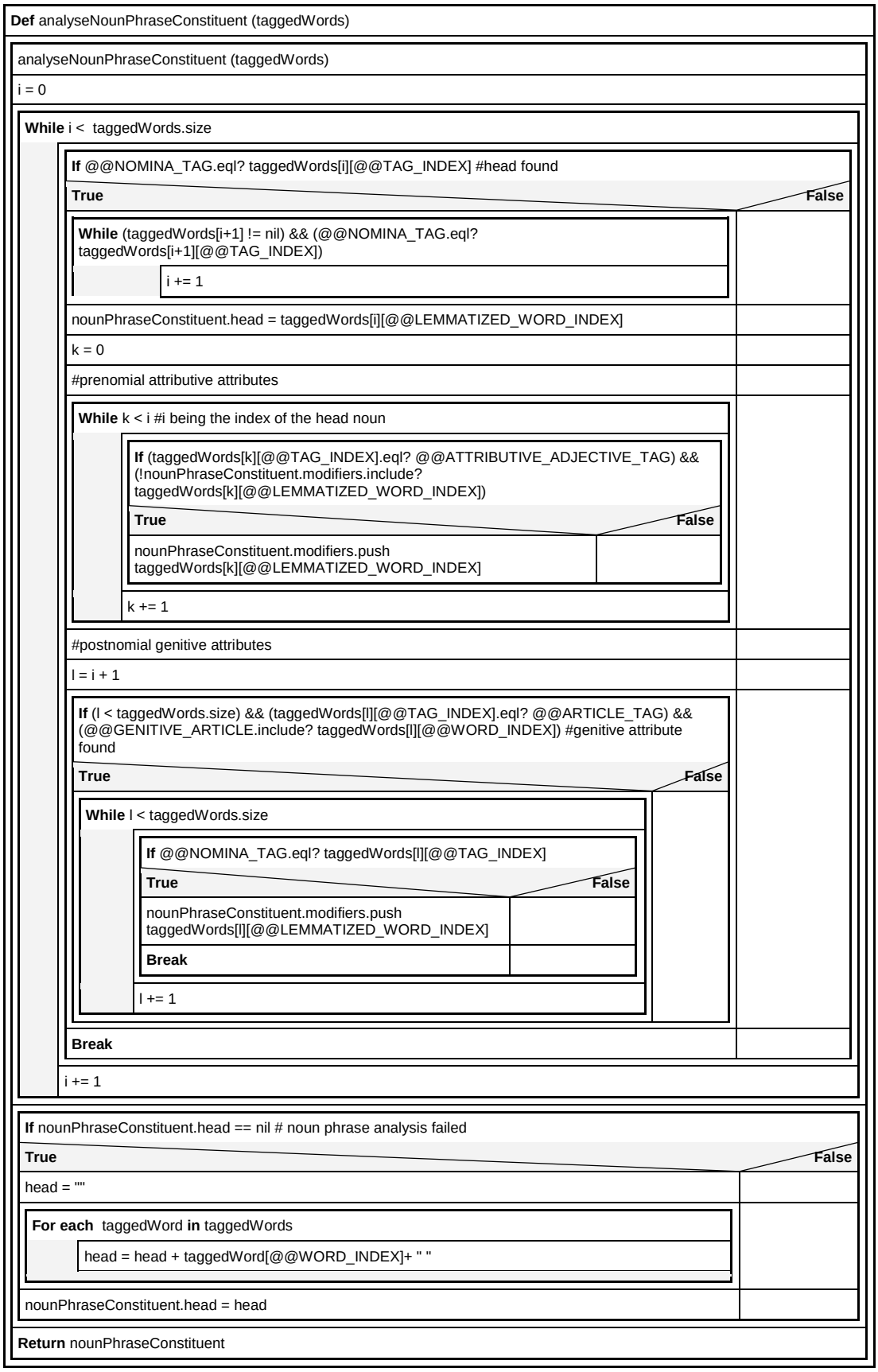

TABELLE 4. STRUKTOGRAMM DER ANALYSE DER DIE NOMINALPHRASE KONSTITUIERENDEN ALTERNATIVEN NOMINALPHRASENKOMPONENTEN

Bei dieser weiteren Analyse wird die Nominalphrase bzw. Nominalphrasenkomponente in den die Hauptbedeutung der Nominalphrase darstellenden Kopf und diesen genauer charakterisierende Bestandteile, im Folgenden Modifikatoren genannt, zerlegt. Der Kopf ist dabei das in der Nominalphrase als erstes auftretende vollwertige Nomen in lemmatisierter Form. Die pränominalen Adjektiv- oder Substantivattribute sowie postnominale Genitivattribute stellen in ihrer lemmatisierten Grundform die Modifikatoren dar. Sollte bei dieser Analyse kein Nominalphrasenkopf gefunden worden sein (z.B. durch fehlerhaftes POS-Tagging), wird für die weitere Verarbeitung einfach die gesamte Nominalphrase in ihrer ursprünglichen Form als Kopf angenommen, um dennoch einen, wenn auch nur auf der nicht-analysierten Zeichenkette beruhenden, terminologischen Ähnlichkeitsvergleich durchführen zu können.[32]

Als letzter Schritt in der Nominalphrasenanalyse werden Komposita in einer abgekürzten Form (z.B. der deutsche Name „Orts- oder Flurname“ der Klasse *E48 Place Name* (*E48 Orts- oder Flurname*)) normalisiert, d.h. der mit Bindestrich endende Bestandteil in Grundform gebracht und anstatt fehlerhaft als Kopf einer eigenen Nominalphrasenkonstituente aufgefasst zu werden, stattdessen als Modifikator dem eigentlichen Kopf der Nominalphrase zugeordnet.

An die Nominalphrasenanalyse anschließend wird eine Kompositazerlegung der zuvor bestimmten Nominalphrasenköpfe in atomare Wortbestandteile durchgeführt. Dieser Schritt ist gerade im Hinblick auf die sinnvolle Verwendung eines Synonymmengen definierenden Thesaurus vonnöten, da aufgrund der unbeschränkten Vielzahl möglicher Kombinationen nur wenige Komposita als feste Begriffe Eingang in Wörterbücher finden. Bei dieser Kompositazerlegung wird nur der letzte Bestandteil des jeweiligen Kompositums weiterhin als den die Hauptbedeutung tragenden Kopf betrachtet, die anderen Wortbestandteile werden den Modifikatoren zugeordnet.

Da nicht die durch die Nominalphrasenkonstituenten bezeichneten Teilkonzepte aus CRM und Kor miteinander verglichen werden sollen, sondern deren Gesamtkonzepte, werden die zuvor ermittelten Nominalphrasenköpfe sowie deren Modifikatoren in jeweils einer Menge zusammengefasst.

Darauf folgend wird für jeden auf seine Grundform zurückgeführten Nominalphrasenkopf in der Menge der Nominalphrasenköpfe die Menge seiner in

[32] So taggt der für die Implementierung verwendete *TreeTagger* beispielsweise vom CRM als Bezeichner verwendete substantivierte Adjektive wie das Wort „Künstliches“ des Bezeichners der Klasse *E71 Man-Made Thing* (*E71 Künstliches*) fehlerhaft als Adjektiv. Siehe auch Kapitel 5.5.

einem eigens für die Anwendung erstellten Thesaurus definierten Synonyme ermittelt. Der Thesaurus beschränkt sich dabei auf Synonyme der die Hauptbedeutung der Bezeichner der CRM-Klassen tragenden Nomen, da ja nur paarweise Nomen aus dem CRM und Kor miteinander verglichen werden müssen, und daher eine Beschränkung auf die bekannten Bezeichner des CRMs ausreichend ist.

Die oben beschriebene linguistische Analyse geht wie bereits erwähnt der eigentlichen, auf Edit-Distanz-Wert basierenden Ähnlichkeitsberechnung eines Kor-Entitätstyps mit jeder der Klassen des CRM voraus.

Um die vom Nutzer angestoßene Ähnlichkeitsberechnung eines Kor-Entitätstyps performanter zu gestalten, werden die vorgestellte Nominalphrasenanalyse und Kompositazerlegung der Bezeichner der statischen CRM-Klassen bereits beim ersten Laden des CRMs (siehe Kapitel 5.1.1) durchgeführt und deren Ergebnisse persistiert. So werden diese für die weitere Anwendung ohne großen zusätzlichen Rechenaufwand verfügbar gemacht. Die Analyse der dynamischen, sich in ihrer Menge sowie auch Bezeichnung prinzipiell ändernden Kor-Entitätsklassen wird bei jedem Matching-Vorgang für den jeweiligen Kor-Entitätstyp neu angestoßen.

Beim terminologischen Ähnlichkeitsvergleich eines Kor-Entitätstyps mit den Klassen des CRMs werden dann für jede Korrelation zwischen Kor-Entitätstyp und jeweiliger Klasse ein Edit-Distanz-Wert ermittelt und die CRM-Klassen nach höchstem erzieltem Ähnlichkeitswert absteigend in einer Liste geordnet.

Die Ordnung kommt dabei in zwei Schritten zustande. In einem ersten Schritt werden die in den aus den Köpfen der die Bezeichner der Elemente konstituierenden Nominalphrasen ermittelten Synonymmengen von Kor-Entitätstyp und jeweiliger CRM-Klasse enthaltenen Synonyme paarweise miteinander verglichen und die Klassen nach dem am besten beim Synonymvergleich erzielten Edit-Distanz-Wert geordnet. In einem zweiten Schritt werden die Modifikatoren ähnlich wie im Falle der Synonyme paarweise miteinander verglichen. Auch hier dient der beste erzielte Edit-Distanz-Wert der Ordnung der CRM-Klassen. Der Edit-Distanz-Wert des Modifikatorenvergleichs ist allerdings nur in zweiter Instanz bei der Ordnung der CRM-Klassen ausschlaggebend, d.h. falls mehrere CRM-Klassen denselben besten Edit-Distanzwert beim Vergleich der Synonyme ihrer ihre Hauptbedeutung definierenden Nominalphrasenköpfe erreicht haben.

5.4.2 Matching von Kor-Relationen und CRM-Property-Pfaden

Das oben beschriebene schrittweise Konstituieren von „eigentlichen“ Relationen semantisch entsprechenden Property-Pfaden ist mühselig. Aus diesem Grund wurde versucht eine Funktion zu implementieren, die direkt einen Pfad von der mit dem Domain-Entitätstypen der „eigentlichen“ Relation gemappten CRM-Klasse zu der mit dem Range-Entitätstypen der „eigentlichen“ Relation gemappten CRM-Klasse berechnet. Die Beobachtung ist folgende: In der Regel kann eine Klasse des CRMs durch maximal zwei Properties von einer anderen Klasse des CRMs aus erreicht werden. Auch die den „eigentlichen“ Relationen entsprechenden Property-Pfade lassen sich in der Regel durch nur wenige Properties konstituieren, so dass es sich hier meist um kürzeste Pfade handelt. Die eigentliche Variabilität der Property-Pfade liegt damit nicht so sehr in ihrer Länge als in der Art der sie konstituierenden Properties.

Def calculateShortestPath (actualRelation, crmProperties, domainClass)		
For crmProperty **in** crmProperties		
crmProperty.similarity = 1 - (KorCrmMatching::EditDistanceCalculator.calculateNormalizedLevenshteinDistance crmProperty.label, actualRelation.relation.namel)		
inheritanceFreeCrmProperties = KorCrmMatching::ShortestPathCalculator.deriveInheritanceFreePropertyGraph crmProperties		
rangeClass = actualRelation.range.crmClass		
shortestPathTree = KorCrmMatching::ShortestPathCalculator.dijkstra inheritanceFreeCrmProperties, domainClass, rangeClass		
For each shortestPathProperty **in** shortestPathTree		
If shortestPathProperty.end == **True**		
True	**False**	
targetClass = shortestPathProperty.actualDomain	/	
Break	/	
shortestPathTree.delete shortestPathProperty		
shortestPath = KorCrmMatching::ShortestPathCalculator.calculateShortestPathFromTargetToSource shortestPathTree, domainClass, targetClass		
shortestPath.reverse!		
Return shortestPath		

TABELLE 5. STRUKTOGRAMM DER BERECHNUNG KÜRZESTER WEGE ZWISCHEN STARTKLASSE UND ZIELKLASSE

Bei einem Mapping-Vorgang für eine „eigentliche“ Relation (Tabelle 5) wird in einem ersten Schritt allen Properties ein auf dem paarweisen terminologischen Vergleich ihrer Namen mit dem Namen der „eigentlichen“ Relation beruhender Ähnlichkeitswert zugeordnet. Da sich die Analyse der die Namen der Relationen bzw. Properties konstituierenden deutschen Verbalphrasen sehr schwierig gestaltet, beruht der

terminologische Ähnlichkeitsvergleich in der aktuellen Version des Tools nur auf der Berechnung der Edit-Distanz zwischen den die Namen repräsentierenden Zeichenketten[33].

Zur Verwendung bekannter Graph-Traversierungs-Algorithmen wird in einem nächsten Schritt der den CRM-Graph repräsentierende Graph aus Programmobjekten, der implizit die Konzepte der Vererbung und Polymorphismus zwischen Klassen des CRMs enthält, in einen Objektgraph überführt, der diese Konzepte expliziert. Dies bedeutet, dass jede im CRM definierte Property, die für ihre definierten Domain- und Range-Klassen als auch für deren Subklassen gilt, im expliziten Graphen in jeweils eine Property zwischen jeder möglichen Kombination der Domain- und Range-Klassen und deren Subklassen aufgespaltet wird.

In einem nächsten Schritt wird mittels Dijkstras Single-Pair-Shortest-Path-Algorithmus [Dijkstra 1959] solange kürzeste Wege von der Start-Klasse zu anderen Klassen im expliziten CRM-Graph berechnet, bis die Ziel-Klasse gefunden wurde. Die Dijkstras Algorithmus implementierende Methode liefert einen Baum kürzester Wege von der Start-Klasse bis zur Ziel-Klasse sowie zu allen Klassen, deren Wege kürzer sind zurück. Da der Property-Pfad auch konstituiert ist, sobald eine Unterklasse der Zielklasse erreicht worden ist, wird die den Algorithmus beendende, die letzte Property des Pfades konstituierende Property zusätzlich als solche ausgezeichnet.

In einem letzten Schritt wird der kürzeste Weg von Start- zu Ziel-Klasse bzw. deren Unterklasse im Baum kürzester Wege berechnet.

5.5 Bewertung verwendeter Technologien und Werkzeuge

Folgend soll eine kurze Bewertung der zur Realisierung des linguistischen Matchings verwendeten Werkzeuge und Technologien versucht werden. Wegen der hohen Komplexität sprachlicher Phänomene musste besonders hier auf spezielle Werkzeuge zurückgegriffen werden. Die Effektivität des für den KCM verwendeten terminologisch/linguistischen Matching-Ansatzes ist daher in hohem Maße von der Effektivität dieser Werkzeuge abhängig, auf denen er aufbaut.

[33] Die Ähnlichkeitswerte sind daher leider aktuell so gut wie zufällig.

Bei Verwendung des POS-Taggers *TreeTagger* fiel auf, dass Wordklassen nicht immer korrekt ausgezeichnet wurden. So wurden beispielsweise die im CRM als deutsche Bezeichner für Klassen gebrauchten substantivierte Adjektive bzw. Partizipien nicht als Nomen, sondern als Adjektive ausgezeichnet. Um diesen Sachverhalt zu korrigieren mussten spezielle Sonderfälle bei der Implementierung des Ansatzes beachtet werden. Auch wäre es nützlich gewesen zusätzlich zur Wortklasse (und der lemmatisierten Grundform) der den Bezeichner konstituierenden Wörter auch deren grammatischen Kasus zu erfahren. Da in einer Nominalphrase der die Hauptbedeutung der Phrase tragende Kopf stets im Nominativ vorkommt, hätte diese Information eine Analyse komplexerer Nominalphrasenstrukturen als die für den KCM letztlich geforderte vereinfachte Nominalphrasenstruktur ermöglicht.

Auch bei der Verwendung des zur Zerlegung deutscher Komposita verwendeten Tools *jWordSplitter* kam es zu Fehlern. So wurden ähnlich des bei Lemmatizern bekannten Phänomens des Under- und Overrstemmings manche Wörter, die Komposita darstellten, nicht zerlegt (z.B. „Gruppenauflösung“), während andere nicht Komposita darstellende Wörter zerlegt wurden (z.B. „Zustandsfeststellung“ -> „Zustand“ – „fest“ – „stellung“).

Der für die Anwendung speziell erstellte Thesaurus ist in seiner Effektivität besonders an die im oben skizzierten Ansatz seinem Einsatz vorausgehenden Verarbeitungsschritte gebunden. Da nur die in *openthesaurus.de* aufgeführten Grundformen, der tatsächlich in den Bezeichnern der CRM-Klassen vorkommenden, im Falle von Komposita korrekt zerlegten Nominalphrasenköpfe in den Thesaurus aufgenommen wurden, ist die korrekte automatische Erkennung von Nominalphrasenköpfen, die Ermittlung deren Grundform sowie die korrekte Zerlegung von Komposita eine notwendige Voraussetzung für den effektiven Einsatz des Thesaurus. Eine zusätzliche Verbesserung der Effektivität des Thesaurus könnte eventuell erzielt werden, indem zusätzlich zu den hier verwendeten Synonymen auch Hyper- und Hyponyme mit in den Thesaurus aufgenommen und diese Abhängigkeiten zur Bestimmung der Ähnlichkeit von Bezeichnern zusätzlich genutzt werden würden. Insbesondere besteht bei der Verwendung von Hyper- und Hyponymen die Hoffnung, dass die Kluft unterschiedlicher Abstraktheitsgrade zwischen den Konzepten des abstrakten CRMs und des konkreteren Datenmodells der Frankfurter ConedaKor-Installation - zu dem wohl für einen bestimmten Anwendungsbereich konzipierte Datenmodelle in der Regel tendieren - überbrückt werden könnte.

6 Evaluierung

6.1 Evaluierungsmethode

6.1.1 Evaluierungsmethode des Mappings zwischen Kor-Entitätstypen und CRM-Klassen

Für jeden Kor-Entitätstyp ist das Resultat des in Kapitel 5.4.1 beschriebenen terminologisch-linguistischen Matching-Ansatzes also eine nach lexikalischer Ähnlichkeit geordnete Liste der Klassen des CRM.

Zur Evaluierung der Effektivität des Matchings wurde zunächst vom Domänenexperten manuell ein Referenzmapping erstellt, das die relevanten bzw. korrekten Korrespondenzen zwischen Kor-Entitätstypen und CRM-Klassen enthält und mit dem die vom terminologisch-linguistischen Matching-Ansatzes berechneten geordneten Trefferlisten verglichen werden sollen (Tabelle 6).

Kor-Entitätstyp	**Entsprechende CRM-Klasse**
Medium	Informationsgegenstand
Ort	Ort
Institution	Juristische Person
Person	Person
Literatur	Dokument
Werk	Hergestelltes
Text	Sprachlicher Gegenstand
Personengruppe	Menschliche Gruppe
Ausstellung	Ereignis
Embedded Medium	Informationsgegenstand
Name	Benennung
Aktiv	Handlung
Baubeginn	Daseinsbeginn
Bauzeit	Bearbeitung
Datierung	Daseinsbeginn

Geburtsdatum	Geburt
Geburtsjahr	Geburt
Gründung	Gruppenbildung
Lebensdaten	Phase
Sterbejahr	Tod
Todesjahr	Tod
Wirkungsdaten	Handlung
Wirkungszeit	Handlung
Zeitraum	Phase

TABELLE 6. REFERENZMAPPING ZWISCHEN KOR-ENTITÄTSTYPEN UND CRM-KLASSEN

Die berechnete, geordnete Liste von CRM-Klassen, beinhaltet immer alle existierenden CRM-Klassen, so dass im Gegensatz zur im Information Retrieval geläufigen Recall- und Precision-Berechnung, die Kennwerte für die Vollständigkeit bzw. Genauigkeit der in Abhängigkeit von einer Anfrage hin berechneten Menge relevanter Dokumente (hier: Namen der CRM-Klassen) berechnen, nicht das Finden der Korrespondenz überhaupt, sondern deren Rang in der geordneten Liste von Bedeutung ist.

Zu diesem Zwecke soll der „Average Precision"-Kennwert verwendet werden, der die durchschnittliche Genauigkeit einer geordneten Trefferliste ausdrückt:

$$AveP = \sum_{r=1}^{N} \frac{P(r) \times rel(r)}{REL}$$

, wobei $P(r)$ die Genauigkeit bei der Betrachtung der obersten r Element der Liste ist, $rel(r) = 1$ ist, falls die Korrespondenz an der Stelle r der Referenzkorrespondenz entspricht, $rel(r) = 0$ ist, falls dem nicht so ist, REL die Anzahl der relevanten Korrespondenzen und N die Anzahl der Korrespondenzen in der Liste ist.

Der Precision-Wert berechnet sich dabei als:

$$P = \frac{GEF \cap REL}{GEF}$$

, wobei GEF die Menge der gefundenen CRM-Klassen und REL die Menge der relevanten CRM-Klassen in Bezug auf den zu mappenden Kor-Entitätstyp darstellt.

Über diesen „Average Precision"-Kennwert für jedes Mapping zwischen Entitätstyp und Klasse soll dann noch einmal ein Durchschnitt für alle Mappings gebildet werden:

$$MAP = \sum_{q=1}^{Q} \frac{AveP(q)}{Q}$$

, wobei q das jeweils betrachtet Matching und Q die Anzahl der Matchings insgesamt ist.

Zusätzlich sollen die Ränge der im Referenzmapping gemappten CRM-Klassen für jeden Kor-Entitätstyp in den geordneten Trefferlisten ermittelt und der durchschnittliche Rang über alle Entitätstypen gebildet werden.

Bei der Evaluierung wurde sich bewusst gegen speziell für Ontologie/Schema-Mapping vorgeschlagene generalisierte, semantische Evaluierungsmaße wie sie beispielsweise in [Euzenat 2007] vorgeschlagen werden entschieden. Dies liegt daran, dass in der hier durchgeführten Evaluierung in erster Linie die Effektivität der auf terminologisch-linguistischen Ansätzen beruhende Matching-Methode und nicht so sehr das resultierende Mapping-Alignment an sich evaluiert werden soll. Die beispielsweise in [Euzenat 2007] vorgeschlagene Erweiterung der Menge relevanter Korrespondenzen (*α-Konsequenz*) auf Grundlage ihrer semantischen Abhängigkeiten wie beispielsweise Teilmengenbeziehungen wäre in Bezug auf die ausschließlich auf lexikalischer Ähnlichkeit beruhende Matching-Methode in den allermeisten Fällen völlig zufällig.

6.1.2 Evaluierungsmethode des Mappings zwischen Kor-Relationen und Property-Pfaden im CRM

Da wie bereits angesprochen die Grundlage des Algorithmus, d.h. die Gewichtung der Properties in Form von lexikalischen Ähnlichkeitswerten zwischen den Namen von Relationen und Properties, aufgrund der fehlenden Analyse der die Namen konstituierenden Verbalphrasen in der aktuellen Version nahezu arbiträr ist, ist die

Evaluierung des Verfahrens nur von wenig Interesse.[34] Von einer Evaluierung dieses Ansatzes wird daher abgesehen.

Nichtsdestotrotz scheint die Verwendung eines Single-Pair-Shortest-Path-Algorithmus ein vielversprechendes Mittel zum Matching von Relationen und Property-Pfaden im CRM zu sein, insofern denn eine verlässliche Berechnung von lexikalischen Ähnlichkeitswerten zwischen den Namen der zu mappenden Elemente zur Verfügung steht. Dies war leider im Rahmen dieser Arbeit nicht mehr realisierbar.

6.2 Evaluierungsergebnisse

Die Ergebnisse der Evaluierung der Matching-Methode sind im Anhang in Abbildung 27 und Abbildung 28, sowie in Tabelle 7 dargestellt. Zur genaueren Evaluierung der Effektivität der einzelnen, die finale Matching-Methode (M4) konstituierenden Haupt-Verarbeitungsschritte wurde diese in zunehmend komplexer werdende, die schrittweise Evolution der finalen Methode darstellende Zwischenschritte aufgebrochen (M1-M3).

Bei der mit M0 bezeichnete „Methode“ werden keine terminologischen oder sonstigen Matching-Ansätze verfolgt. Hier sind die Ergebnisse ausschließlich nach Codenummer der CRM-Klassen aufsteigend geordnet und damit in Bezug auf die terminologische Ähnlichkeit der zu mappenden Elemente quasi zufällig. Die von dieser „Methode“ erzielten Ergebnisse dienen als Referenz bei der Evaluierung der realisierten terminologisch-linguistischen Matching-Ansätze.

Die Methode M1 verwendet zur Berechnung des Ähnlichkeitswertes einer Korrespondenz ausschließlich den durch die Levenshtein-Metrik realisierten Edit-Distanz-Wert zwischen den kompletten, den Namen der zu mappenden Elemente repräsentierenden Zeichenketten.

M2 analysiert zusätzlich vor Berechnung des Edit-Distanz-Wertes die Struktur der den Namen der zu mappenden Elemente darstellenden Nominalphrasen. Die Nominalphrasen werden dabei in Nominalphrasenkopf und andere, den Kopf näher beschreibende Nominalphrasenbestandteile, hier Modifikatoren genannt, zerlegt. In einem ersten Schritt werden die Ähnlichkeitswerte zwischen zu vergleichenden Nominalphrasenköpfen berechnet. Erst in einem zweiten Schritt werden die

[34] So wurde im zur Evaluierung durchgeführten Versuch nicht ein einziger einer Kor-Relation entsprechender Property-Pfad berechnet.

Ähnlichkeitswerte der Modifikatoren berechnet, so dass bei gleichen Ähnlichkeitswerten der Köpfe die zugehörigen CRM-Klassen weiter nach Ähnlichkeit der Modifikatoren geordnet werden können.

M3 führt zusätzlich nach der Nominalphrasenanalyse eine Zerlegung der Nominalphrasenkopfkomposita durch. Dabei wird nur das letzte eigenständige Wort des Kompositums weiterhin als Kopf beibehalten, die anderen Kompositabestandteile werden der Menge der Modifikatoren zugeordnet. Anschließend wird ein wie bei M2 beschriebener Edit-Distanz-Vergleich durchgeführt.

M4 ist schließlich die finale Methode, die in dieser Form auch vom KCM verwendet wird. Hier wird zusätzlich zum für M3 beschriebenen Vorgehen nach Nominalphrasenanalyse und Kompositazerlegung ein Synonyme verzeichnender Thesaurus eingesetzt. Beim Edit-Distanz-Vergleich werden dann die Ähnlichkeitswerte zwischen Synonymen der Nominalphrasenköpfe zweier zu mappender Elemente als Ähnlichkeitswerte für den gesamten Nominalphrasenkopfvergleich verwendet, deren Werte die höchsten sind.

Schon die Einführung der einfachen, auf der Levenshtein-Metrik beruhenden Edit-Distanz-Berechnung (M1) führt zu einer bedeutenden Präzision der geordneten Ergebnisliste. Der MAP-Wert erhöht sich von 0,066927248 im Falle ohne Matching (M0) auf 0,323739623 im Falle des Anwendens der Edit-Distanz-Berechnung (M1). Dies entspricht einer Verbesserung der mit dem Kor-Entitätstyp korrespondierenden CRM-Klasse vom durchschnittlich ca. 38. (M0) auf den durchschnittlich ca. 28. (M1) Rang in der geordneten Trefferliste.

Durch Anwendung der Nominalphrasenanalyse und der daraus resultierenden Strukturierung der den Namen der zu mappenden Elemente repräsentierende Nominalphrase in Hauptbedeutung, d.h. Nominalphrasenkopf, und die Hauptbedeutung modifizierende Nominalphrasenbestandteile, d.h. Modifikatoren, (M2) kommt es bei der stufenweisen Edit-Distanz-Berechnung - zunächst der Köpfe und erst in zweiter Instanz der Modifikatoren - zu einer weiteren Verbesserung des MAP-Wertes von 0,323739623 (M1) auf 0,335764218 (M2). Der Durchschnittsrang der korrespondierenden CRM-Klasse sinkt allerdings von ca. 28 (M1) auf ca. 31 (M2).

Bei der zusätzlichen Zerlegung der Nominalphrasenkopfkomposita und der Beibehaltung des ausschließlich letzten, d.h. in der Regel die Hauptbedeutung des Kompositums darstellenden Kompositumbestandteils als Kopf der den Namen des Elements darstellenden Nominalphrase und der Zuordnung der anderen Komositumkomponenten zu den die Bedeutung des Kopf modifizierenden Modifikatoren (M3) kommt es zunächst zu einem Einbruch des MAP-Wertes von

0,335764218 (M2) auf 0,24319342 (M3) sowie einer Verschlechterung des Durchschnittsranges der Treffer von ca. 31 (M2) auf ca. 34 (M3).

Diese Zerlegung der Komposita ist allerdings eine notwendige Voraussetzung, um mit einem Thesaurus arbeiten zu können, der Synonymmengen eigenständiger Wörter verzeichnet. Bei der Nutzung eines solchen Thesaurus (M4), der sich wie oben beschrieben zusammensetzt, ergibt sich dann auch wieder eine Verbesserung des MAP-Wertes von 0,24319342 (M3) auf 0,310647556 (M4). Der MAP-Wert der Matching-Methode bleibt allerdings leicht hinter den MAP-Werten für die Matching-Methoden, die nur auf einer Edit-Distanz-Berechnung bzw. Nominalphrasenanalyse und darauf folgender Edit-Distanz-Berechnung (0,323739623 (M1) bzw. 0,335764218 (M2)) beruhen, zurück. Wirft man allerdings einen Blick auf den Durchschnittsrang der Treffer in der Trefferliste liefert die mit einem Synonymthesaurus, auf Nomainalphrasenanalyse und Kompositazerlegung basierende Matching-Methode (M4) mit einem Durchschnittsrang von 23 das im Vergleich mit den zuvor genannten Matching-Methoden beste Ergebnis.

Erzielt Matching-Methode M4 global die besten Ergebnisse und liefert durchschnittlich die präzisesten Treffer, fallen in der Einzelbetrachtung einige Fälle auf, in denen die primitiveren Matching-Methoden, insbesondere M1 und M2, präziser sind.

Dies trifft beispielsweise für die im Versuch betrachteten Daten für die Kor-Entitätsklassen mit den Bezeichnern „Bauzeit“, „Gründung“, „Geburtsdatum“, Geburtsjahr“ und „Todesjahr“ zu. Ist dies in den Fällen der Bezeichner „Bauzeit“ und Gründung“ durch die relative, wenn auch äußerst geringe und damit praktisch zufällige Ähnlichkeit mit den Bezeichnern der semantisch korrespondierenden CRM-Klassen „Bearbeitung“ bzw. „Gruppenbildung“ zu erklären, so kann für die übrigen oben genannten Bezeichner eine Erklärung gefunden werden, die die Schwächen der Matching-Methode M4 aufzeigt.

Wird durch Matching-Methode M1 und M2 die mit den Kor-Entitätsklassen mit den Bezeichnern „Geburtsdatum“, „Geburtsjahr“ und „Todesjahr“ korrespondierenden CRM-Klassen „Geburt“ bzw. „Tod“ als ähnlichste Klassen berechnet, so ist dies bei Matching-Methode M4 nicht mehr der Fall. Schuld daran ist die Kompositazerlegung, die für das Matching in erster Linie die letzte Komponente des Kompositums, d.h. „Datum“ bzw. „Jahr“ betrachtet während in M1 und M2 die in M4 als Modifikatoren erst in zweiter Instanz zur Ähnlichkeitsberechnung herangezogenen Komposita-Komponenten „Geburt“ bzw. „Tod“ mit in die Ähnlichkeitsberechnung erster Ebene mit einfließt. Eine Verbesserung könnte hier eventuell erzielt werden, indem die Ähnlichkeiten der zu vergleichenden Nominalphrasenköpfe vor und nach der

Kompositazerlegung verglichen werden und der höhere Wert als Ähnlichkeitsmaß der Korrelation angenommen wird.

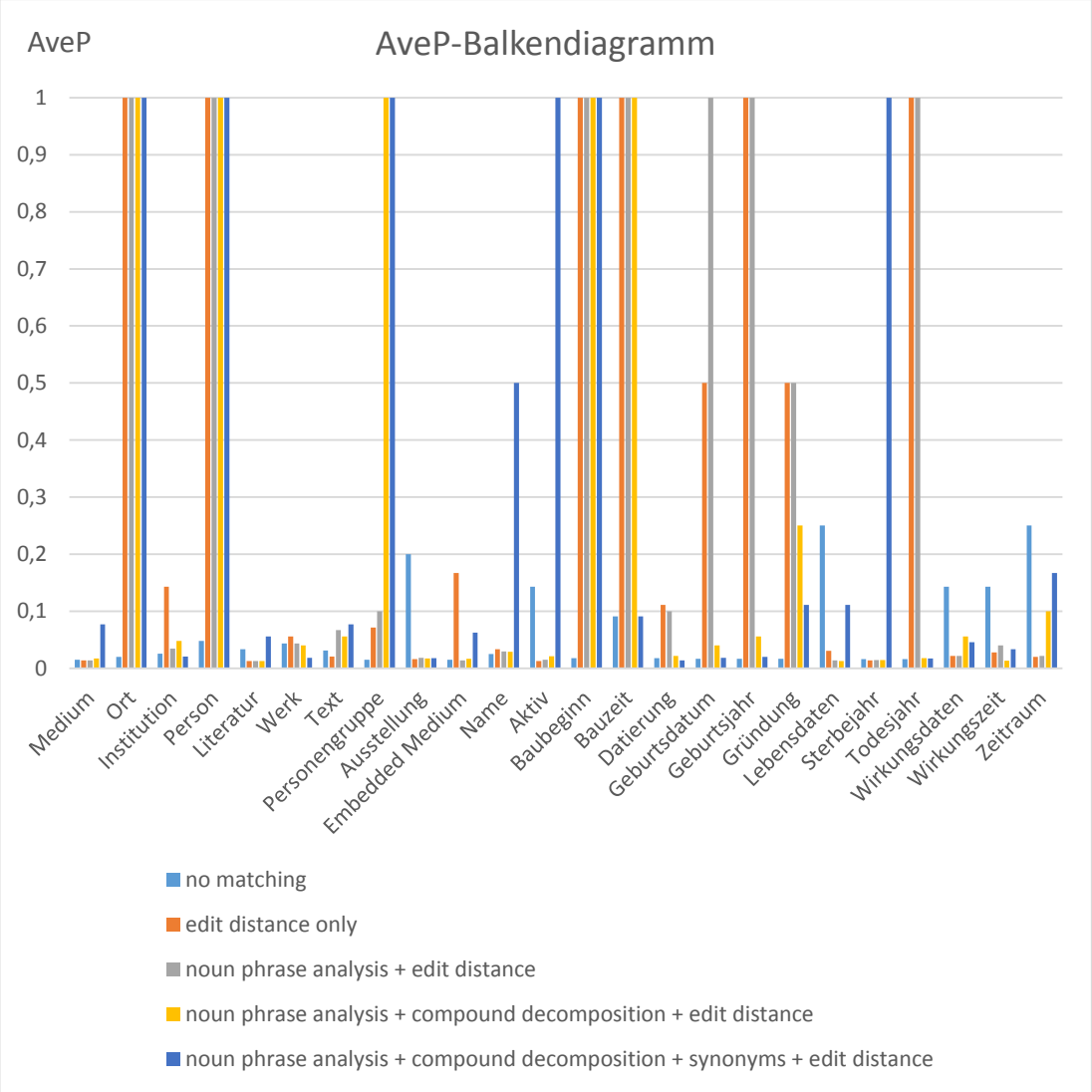

TABELLE 7. BALKENDIAGRAMM DER AVEP-ERGEBNISSE FÜR ALLE GEMAPPTEN KOR-ENTITÄTSTYPEN FÜR JEWEILIGE MATCHING-METHODE

6.3 Diskussion der Evaluierung

Die Diskrepanz zwischen MAP-Werten der Matching-Methoden und dem Durchschnittsrang des Treffers in den von diesen berechneten geordneten Trefferlisten kommt dadurch zustande, dass die AveP-Werte hyperbolisch abnehmen und somit in der Trefferliste besonders hoch angesiedelte Treffer mit einem ebenfalls besonders hohen AveP-Wert in den MAP-Wert einfließen, während der Wert der Durchschnittsränge nur linear wächst. So entspricht einem Treffer mit Rang 2 in der Trefferliste ein AveP-Wert von 0,5 während einem Treffer mit Rang 4 ein AveP-Wert von 0,25 entspricht. Dies bedeutet bereits einen Unterschied von Faktor 4 im Vergleich der beiden Verhältnisse zwischen AveP-Wert und Rang.

Bei dem nahezu gleichen MAP-Wert der drei Matching-Methoden, von der eine nur die Edit-Distanz berechnet (M1), die zweite zusätzlich eine Nominalphrasenanalyse durchführt (M2) und die letzte schließlich zusätzlich auch eine Kompositazerlegung durchführt und Synonymsets verwendet (M4), kommt der im Vergleich zu M1 (ca. 28) und M2 (ca. 31) besonders gute Durchschnittsrang von M4 (ca. 23) also dadurch zustande, dass zwar im Vergleich zu M1 und M2 nicht mehr sehr hohe Trefferränge erzielt werden, sondern dass insgesamt im Vergleich zu M1 und M2 weniger sehr niedrige Trefferränge erzielt werden.

Da der realisierte terminologisch-linguistische Mapping-Ansatz letztlich allerdings darauf ausgelegt ist Wortpaare zu finden, die terminologisch möglichst gleich sind, ist das Vorkommen der Treffer in der Mitte der geordneten Liste wohl eher zufällig. Der MAP-Wert ist daher dem ermittelten Durchschnittsrang in Bezug auf seine Aussagekraft über die Effektivität des Matching-Ansatzes vorzuziehen.

Dies bedeutet aber auch, dass die durchgeführten linguistischen Ansätze - Nominalphrasenanalyse, Kompositazerlegung, Verwendung von Synonymengen - im Vergleich zu einem einfachen terminologischen, auf den reinen Zeichenketten der Namen der zu vergleichenden Elemente beruhenden Ansatz zu keiner nennenswerten Verbesserung der Präzision des automatischen Mappings geführt haben.

Allerdings besteht die Erwartung, dass bei der weiteren Verbesserung des realisierten Ansatzes - vor allem der größeren Bewertung der im aktuellen Ansatz wenig Gewicht gegebenen Modifikatoren – eine Verbesserung der Präzision erreicht werden kann.

7 Diskussion und Ausblick

Im Rahmen dieser Arbeit wurde ein Mapping-Werkzeug zur Lösung des in Kapitel 2 beschriebenen Problems der semantischen Datenheterogenität zwischen den zu integrierenden ConedaKor-Datenbanken realisiert. Dabei wurden auf die in Kapitel 3 vorgestellten, in aktueller Forschungsliteratur dargestellten Lösungsansätze zurückgegriffen, die in Kapitel 4 im Hinblick auf das konkrete Problem des Abbildens der relationalen Daten der ConedaKor-Datenbanken auf die interoperable Ontologie des CIDOC CRM diskutiert wurden. Der aus der Diskussion folgende, im Rahmen dieser Arbeit realisierte Lösungsansatz wurde in Kapitel 5 in seiner softwaretechnischen Umsetzung beschrieben. In Kapitel 6 schließlich wurde das Mapping-Werkzeug in Bezug auf realisierte automatische Matching-Ansätze evaluiert. Dabei musste die Evaluierung auf das terminologisch-linguistische-Matching zwischen Schema-Elementen von ConedaKor und CIDOC CRM beschränkt werden.

Nach der Transformation der Datenstrukturen der beiden aufeinander abzubildenden Quellen, ConedaKor und CIDOC CRM, in ein gemeinsames objektorientiertes Datenmodell, erlaubt das realisierte semi-automatische Werkzeug mit graphischer Oberfäche *KCM* die Erstellung persistenter Mappings zwischen Kor-Entitätstypen und CRM-Klassen sowie Kor-Relationen und CRM-Property-Pfaden.

Für das Mapping zwischen Kor-Entitätstypen und CRM-Klassen wurde ein auf terminologisch-linguistischen Methoden basierender automatischer Matching-Ansatz realisiert, der wie seine Evaluierung zeigte durchaus brauchbare, allerdings noch zu optimierende, Ergebnisse liefert.

Zum Mapping von Kor-Relationen mit CRM-Property-Pfaden wurde die Mapping Methode von Nußbaumer et al. in leicht angepasster Form realisiert. Die versuchte automatische Unterstützung dieser Mapping-Methode liefert in ihrer aktuellen Form leider keine brauchbaren Ergebnisse. Da die Verbesserung des Ansatzes im Rahmen dieser Arbeit nicht mehr möglich war, wurde von einer Evaluierung des Ansatzes abgesehen. Dennoch ist - auch wenn der angedachte automatische Mapping-Ansatz im Rahmen dieser Arbeit nicht mehr zufriedenstellend realisiert werden konnte - das rein manuelle Mapping, das in [Nußbaumer et al. 2010] vorgeschlagen wird, durch *KCM* realisiert. Allerdings hat sich hier die von Nußbaumer et al. vorgeschlagene Methode als nicht ausreichend erwiesen, um Konzepte semantisch äquivalent mappen zu können. In mehreren Fällen fanden Kor-Relationen durch einfache, unverzweigte CRM-Property-Pfade semantisch keine Entsprechung. Die Relation *hat Lebensdaten* von Entitäten des Typs *Person* zu Entitäten des Typs *Lebensdaten*

müsste beispielsweise durch zwei Properties bzw. Property-Pfade, nämlich *P98 wurde geboren durch* (*P98 was born*) und *P100 starb* (*P100 died in*) gemappt werden können. Ebenso müsste es möglich sein intermediäre Knoten, d.h. Entitäten von Klassen in Property-Pfaden, in anderen Property-Pfaden, referenzieren zu können. Um semantisch im CRM ausdrücken zu können, dass es sich bei der in der Frankfurter Coneda-Kor-Installation vorgehaltenen Entität *Würzburger Residenz* in den Relationen *Werk ist Auftrag von* zu *Schönborn, Johann Philipp Franz von* und *wurde geschaffen von* zu *Balthasar Neumann* implizit um denselben Bauprozess handelt, müsste die beim Mapping auf einen CRM-Property-Pfad explizit gemachte Entität der Klasse *E12 Herstellung* (*E12 Production*) beim Mapping beider Relationen referenziert werden können.

Im Rückblick auf die Arbeit mit dem CIDOC CRM soll die optimistische Einschätzung (so beispielsweise [Ermert, Gottschewski und Hagedorn-Saupe 2005]) dieser vom ICOM entwickelten Ontologie zur Modellierung musealer Daten kritisch nuanciert werden. So hat sich die ereigniszentrierte Struktur des CIDOC CRM beim Finden von Korrespondenzen insbesondere zu Relationen des eher objektzentrierten Datenmodells der Frankfurter ConedaKor-Datenbank wenig intuitiv und umständlich erwiesen. Bei der apriorischen Annahme, dass es sich bei musealen Datenstrukturen in der Regel um objektzentrierte Strukturen handelt, die die realweltliche objektzentrierte Sammlungspraxis von Museen wiederspiegeln, ist es durchaus verwunderlich, dass der ICOM ein ereigniszentriertes Modell, mit dem Anspruch als interoperabler Standard für museale Dokumentationsstrukturen zu dienen, gewählt hat.

Auch ist allgemein bei der Verwendung von quantitativen Verfahren in der kunsthistorischen Forschung, für die Datenbanken, insbesondere mit großen, integrierten Datenmengen, ein Mittel sind, Vorsicht geboten. Denn hier muss bedacht werden, dass es sich bei kunsthistorischen Daten meist nicht um objektive Forschungsdaten wie jene, die beispielsweise in den Naturwissenschaften gewonnen werden, handelt. So gibt es durchaus unterschiedliche Meinungen von Kunsthistorikern wenn es beispielsweise um die Zuschreibung eines Werkes an einen Künstler handelt. Gerade bei der Integration unterschiedlicher Datenbestände stellt sich daher die Frage, wie mit solchen widersprüchlichen Daten umgegangen werden kann. Mit quantitativen Mitteln gewonnene Einsichten werden wohl in den meisten Fällen auch qualitativ geprüft werden müssen und bieten sich damit weniger zur abschließenden Beantwortung kunsthistorischer Fragestellungen, denn als Mittel zum Aufdecken neuer Fragestellungen an. Die Kunstgeschichte wird wohl auch in Zukunft eine in erster Linie hermeneutische Wissenschaft bleiben.

8 Anhang

	AveP for Matching-Method				
	no matching (M0)	edit distance only (M1)	noun phrase analysis + edit distance (M2)	noun phrase analysis + compound decomposition + edit distance (M3)	noun phrase analysis + compound decomposition + synonyms + edit distance (M4)
Medium	0,014925373	0,013888889	0,01369863	0,016949153	0,076923077
Ort	0,019607843	1	1	1	1
Institution	0,025641026	0,142857143	0,034482759	0,047619048	0,020408163
Person	0,047619048	1	1	1	1
Literatur	0,033333333	0,012658228	0,012345679	0,012820513	0,055555556
Werk	0,043478261	0,055555556	0,043478261	0,04	0,018181818
Text	0,03125	0,020408163	0,066666667	0,055555556	0,076923077
Personengruppe	0,014705882	0,071428571	0,1	1	1
Ausstellung	0,2	0,016129032	0,018181818	0,017241379	0,01754386
Embedded Medium	0,014925373	0,166666667	0,01369863	0,016393443	0,0625
Name	0,025	0,033333333	0,029411765	0,028571429	0,5
Aktiv	0,142857143	0,012658228	0,014925373	0,020833333	1
Baubeginn	0,01754386	1	1	1	1
Bauzeit	0,09090091	1	1	1	0,09090091
Datierung	0,01754386	0,111111111	0,1	0,02173913	0,013888889
Geburtsdatum	0,016393443	0,5	1	0,04	0,018181818
Geburtsjahr	0,016393443	1	1	0,055555556	0,019607843
Gründung	0,016666667	0,5	0,5	0,25	0,111111111
Lebensdaten	0,25	0,03030303	0,013888889	0,012820513	0,111111111
Sterbejahr	0,015873016	0,01369863	0,014084507	0,014285714	1
Todesjahr	0,015873016	1	1	0,01754386	0,017241379
Wirkungsdaten	0,142857143	0,021276596	0,02173913	0,055555556	0,045454545
Wirkungszeit	0,142857143	0,027777778	0,04	0,013157895	0,033333333
Zeitraum	0,25	0,02	0,02173913	0,1	0,166666667
MAP	0,066927248	0,323739623	0,335764218	0,24319342	0,310647556

ABBILDUNG 27. AVERAGE-PRECISION-WERTE DER KORRESPONDENZEN FÜR MATCHING-METHODEN M0 - M4

	Ranks for Matching-Method				
	no matching (M0)	edit distance only (M1)	noun phrase analysis + edit distance (M2)	noun phrase analysis + compound decomposition + edit distance (M3)	noun phrase analysis + compound decomposition + synonyms + edit distance (M4)
Medium	67	72	73	59	13
Ort	51	1	1	1	1
Institution	39	7	29	21	49
Person	21	1	1	1	1
Literatur	30	79	81	78	18
Werk	23	18	23	25	55
Text	32	49	15	18	13
Personengruppe	68	14	10	1	1
Ausstellung	5	62	55	58	57
Embedded Medium	67	6	73	61	16
Name	40	30	34	35	2
Aktiv	7	79	67	48	1
Baubeginn	57	1	1	1	1
Bauzeit	11	1	1	1	11
Datierung	57	9	10	46	72
Geburtsdatum	61	2	1	25	55
Geburtsjahr	61	1	1	18	51
Gründung	60	2	2	4	9
Lebensdaten	4	33	72	78	9
Sterbejahr	63	73	71	70	1
Todesjahr	63	1	1	57	58
Wirkungsdaten	7	47	46	18	22
Wirkungszeit	7	36	25	76	30
Zeitraum	4	50	46	10	6
Average Rank	37,70833333	28,08333333	30,79166667	33,75	23

ABBILDUNG 28. RÄNGE DER KORRESPONDENZEN FÜR MATCHING-METHODEN M0 - M4

9 Literaturverzeichnis

Angles, R., & Gutierrez, C. (2008). Survey of graph database models. ACM Computing Surveys (CSUR), 40(1), 1.

Berners-Lee, T. (1998). Semantic web road map.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific american, 284*(5), 28-37.

Binding, C., May, K., & Tudhope, D. (2008). Semantic interoperability in archaeological datasets: Data mapping and extraction via the CIDOC CRM. In *Research and Advanced Technology for Digital Libraries* (pp. 280-290). Springer Berlin Heidelberg.

Borst, W. N. (1997). *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente.

Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, *1*(1), 9-36.

Choi, N., Song, I. Y., & Han, H. (2006). A survey on ontology mapping. *ACM Sigmod Record*, *35*(3), 34-41.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377-387.

Cohen, W., Ravikumar, P., & Fienberg, S. (2003, August). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation* (Vol. 3, pp. 73-78).

Crofts, N., Doerr, M., Gill, T., Stead, S., & Stiff, M. (2011). Definition of the CIDOC Conceptual Reference Model. Version 5.0. 4 (November 2011). ICOM/CIDOC CRM Special Interest Group.

Cuno, J. (2012). How Art History is failing at the Internet?. *The Daily Dot*. Verfügbar unter http://www.dailydot.com/opinion/art-history-failing-internet/ [24.10.2014].

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269-271.

Doan, A., & Halevy, A. Y. (2005). Semantic integration research in the database community: A brief survey. *AI magazine, 26*(1), 83.

Doerr, M. (2003). The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI magazine, 24*(3), 75.

Doerr, M. (2009). Ontologies for cultural heritage. In *Handbook on Ontologies* (pp. 463-486). Springer Berlin Heidelberg.

Ermert, A., Gottschewski, J., Hagedorn-Saupe, M., Hansen, H. J., Heuchert, R., Saro, C., ... & Stein, R. Das CIDOC Conceptual Reference Model: Eine Hilfe für den Datenaustausch?.

Euzenat, J. (2007, January). Semantic Precision and Recall for Ontology Alignment Evaluation. In *IJCAI* (pp. 348-353).

Ghawi, R., & Cullot, N. (2007, September). Database-to-ontology mapping generation for semantic interoperability. In *VDBL'07 conference, VLDB Endowment ACM* (pp. 1-8).

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, *5*(2), 199-220.

Guarino, N., Oberle, D., & Staab, S. (2009). What is an Ontology?. In *Handbook on ontologies* (pp. 1-17). Springer Berlin Heidelberg.

Heckel, R. (2006). Graph transformation in a nutshell. Electronic notes in theoretical computer science, 148(1), 187-198.

Hoppe, S., & Schelbert, G. (2013). Für ein verstärktes Engagement in den Digital Humanities. Der Arbeitskreis Digitale Kunstgeschichte. *AKMB-news*, *19*(2), 40-42.

Hu, W., Jian, N., Qu, Y., & Wang, Y. (2005, October). Gmo: A graph matching for ontologies. In *Proceedings of K-CAP Workshop on Integrating Ontologies* (pp. 41-48).

Kalfoglou, Y., & Schorlemmer, M. (2003). Ontology mapping: the state of the art. *The knowledge engineering review, 18*(01), 1-31.

Kohle, H. (2013). Digitale Bildwissenschaft. Verfügbar unter http://archiv.ub.uni-heidelberg.de/artdok/2185/1/Kohle_Digitale_Bildwissenschaften_2013 [21.10.2014].

Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data*

Engineering, 2002. Proceedings. 18th International Conference on (pp. 117-128). IEEE.

Nußbaumer, P., Haslhofer, B. (2007). Putting the CIDOC CRM into Practice – Experiences and Challenges. (Technical Report TR - 200). University of Vienna.

Nussbaumer, P., Haslhofer, B., & Klas, W. (2010). Towards Model Implementation Guidelines for the CIDOC Conceptual Reference Model.

Oldman, D., & Labs, C. R. M. (2014). The CIDOC Conceptual Reference Model (CIDOC-CRM): PRIMER.

Otero-Cerdeira, L., Rodríguez-Martínez, F. J., & Gómez-Rodríguez, A. (2015). Ontology matching: A literature review. *Expert Systems with Applications, 42*(2), 949-971.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *the VLDB Journal, 10*(4), 334-350.

Rodriguez, M. A., & Neubauer, P. (2010). Constructions from dots and lines. *Bulletin of the American Society for Information Science and Technology*, *36*(6), 35-41.

Schmid, H. (1994, September). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the international conference on new methods in language processing* (Vol. 12, pp. 44-49).

Schmid, H. (1995). Improvements in part-of-speech tagging with an application to German. In *In Proceedings of the ACL SIGDAT-Workshop*.

Shvaiko, P., & Euzenat, J. (2005). A survey of schema-based matching approaches. In *Journal on Data Semantics IV* (pp. 146-171). Springer Berlin Heidelberg.

Shvaiko, P., & Euzenat, J. (2013). Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, *25*(1), 158-176.

Spanos, D. E., Stavrou, P., & Mitrou, N. (2012). Bringing relational databases into the semantic web: A survey. *Semantic Web, 3*(2), 169-209.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge engineering, 25*(1), 161-197.

Tudhope, D., Binding, C., May, K., & Charno, M. (2013, September). Pattern based mapping and extraction via CIDOC CRM. In *Workshop Practical Experiences with CIDOC CRM and its Extensions (CRMEX 2013), 17th International Conference on Theory and Practice of Digital Libraries (TPDL 2013)* (Vol. 26, pp. 23-36).