HOCHSCHULE
MITTWEIDA
University of Applied Sciences

MASTERARBEIT

Herr
Lukas Jaeckel, B.Sc.

Entwicklung einer Ontologie zur Repräsentation mobiler Kommunikation im Kontext forensischer Untersuchungen

Mittweida, Oktober 2023

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERARBEIT

Entwicklung einer Ontologie zur Repräsentation mobiler Kommunikation im Kontext forensischer Untersuchungen

Autor:
Lukas Jaeckel

Studiengang:
Cybercrime/Cybersecurity

Seminargruppe:
CY20wC-M

Erstprüfer:
Prof. Dr. rer. nat. Dirk Labudde

Zweitprüfer:
Prof. Dr. rer. nat. Michael Spranger

Einreichung:
Mittweida, 20.10.2023

Verteidigung/Bewertung:
Mittweida, 2023

Faculty of **Applied Computer Sciences and Biosciences**

MASTER THESIS

Development of an Ontology to Represent Mobile Communication in the Context of Forensic Investigations

Author:
Lukas Jaeckel

Course of Study:
Cybercrime/Cybersecurity

Seminar Group:
CY20wC-M

First Examiner:
Prof. Dr. rer. nat. Dirk Labudde

Second Examiner:
Prof. Dr. rer. nat. Michael Spranger

Submission:
Mittweida, 20.10.2023

Defense/Evaluation:
Mittweida, 2023

Bibliografische Beschreibung:

Jaeckel, Lukas:
Entwicklung einer Ontologie zur Repräsentation mobiler Kommunikation im Kontext forensischer Untersuchungen. – 2023. – 63 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften, Masterarbeit, 2023.

Referat:

Die vorliegende Arbeit untersucht, wie eine Ontologie mobile Kommunikation für forensische Auswertungen abbilden kann und welche Chancen sich aus dieser Art von Repräsentation ergeben. Prinzipiell stellen Ontologien einen Lösungsansatz für die wachsenden Herausforderungen im Bereich der digitalen Forensik dar. Vor allem die Heterogenität und stark zunehmende Menge der auszuwertenden Daten stellt die Strafverfolgungsbehörden vor Probleme. Forensische Tools unterstützen bei der Extraktion und Analyse von Daten. Allerdings weisen sie in bestimmten Aspekten ihre individuellen Grenzen auf. Ontologien ermöglichen dabei die Interoperabilität zwischen forensischen Tools und somit die Kombination der jeweiligen Vorteile von diesen Tools. Somit können insbesondere (Teil-)Automatisierungen im Ermittlungsprozess realisiert werden, was zur Ersparnis von Zeit und Ressourcen führt. Darüber hinaus lassen sich anhand von Ontologien logische Schlussfolgerungen herleiten und weitere Methoden aus dem Bereich der künstlichen Intelligenz anwenden. Diese Arbeit verwendet die CASE-Ontologie als Grundlage zur Entwicklung einer Ontologie, welche mobile Kommunikation im Kontext forensischer Untersuchungen repräsentiert. Darüber hinaus wird im experimentellen Teil der Arbeit das Datenmodell einer forensischen Plattform zur Auswertung mobiler Kommunikation auf die entworfene Ontologie abgebildet. Zusätzlich wird ein semantischer Webserver prototypisch aufgesetzt, um einen Anwendungstest der Ontologie durchführen zu können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

API . . . . . Application Programming Interface

BSI . . . . . Bundesamt für Sicherheit in der Informationstechnik

CASE . . . . . Cyber-investigation Analysis Standard Expression

DEX . . . . . Digital Evidence Exchange

DFAX . . . . . Digital Forensic Analysis eXpression

DFXML . . . . . Digital Forensics XML

DIALOG . . . . . Digital Investigation Ontology

EFIOSN . . . . . Event-based forensic Integration Ontology for Online Social networks

EMF . . . . . Eclipse Modeling Framework

F-DOS . . . . . Forensic-Driven Ontologies for Smartphones

IRI . . . . . Internationalized Resource Identifier

MoNA . . . . . Mobile Network Analyzer

OWL . . . . . OWL Web Ontology Language

OWL 2 . . . . . OWL 2 Web Ontology Language

ParFor . . . . . Parallax Forensics

RDF . . . . . Resource Description Framework

RDFS . . . . . RDF Schema

UCO . . . . . Unified Cyber Ontology

W3C . . . . . World Wide Web Consortium

WSFO . . . . . Web Services Forensic Ontology

1 Einleitung

1.1 Motivation

Instant-Messaging-Dienste verzeichnen in der heutigen Gesellschaft eine kontinuierlich wachsende Bedeutung. So nutzten im Jahr 2020 etwa 2,91 Milliarden Menschen weltweit derartige Dienste [1]. Ausgehend von dieser Statistik wird ein Anstieg auf 3,51 Milliarden weltweiter Nutzer bis 2025 prognostiziert [1]. Die zunehmende Beliebtheit mobiler Kommunikation führt dazu, dass eine Vielzahl von verschiedenen Messenger-Diensten entwickelt und angeboten wird. Hierbei versucht sich jeder Messenger mit der Entwicklung von möglichst neuartigen Funktionalitäten von der Konkurrenz abzuheben. Dazu werben sie unter anderem mit Sicherheitsfeatures, wie beispielsweise Telegram mit der Verschlüsselung oder dem automatisierten Löschen von Daten [2]. Die weite Verbreitung sowie Sicherheitsfeatures von Messenger-Diensten bieten jedoch auch die Möglichkeit, Straftaten vorzubereiten oder zu begehen. So können beispielsweise Minderjährige kontaktiert werden, um sich ihnen durch Manipulation anzunähern und sexuelle Kontakte anzubahnen. Dieses Vorgehen wird als Cyber-Grooming bezeichnet und ist nach § 176 StGB strafbar [3]. Weitere Beispiele für mögliche Straftaten, welche durch die Nutzung von Messenger-Diensten begangen werden können, sind Verbreitung, Erwerb und Besitz kinder- und jugendpornografischer Inhalte nach §§ 184b und 184c StGB, die Bildung einer kriminellen Vereinigung nach § 129 StGB sowie Straftaten nach Paragraph 29 des Betäubungsmittelgesetzes. Deshalb stehen für Strafverfolgungsbehörden vermehrt Instant-Messaging-Dienste im Fokus ihrer Ermittlungen.

Allerdings stellt die Auswertung von mobilen Endgeräten sowie von darauf installierten Messenger-Diensten die Forensik vor eine Vielzahl von Herausforderungen. Einerseits besitzen die Geräte und Dienste unterschiedlichste Sicherheitsfeatures, welche das Auslesen oder die Wiederherstellung von Daten erschweren. Andererseits befinden sich in der Regel sehr große Mengen von Daten in unterschiedlichsten Formen auf einem einzelnen Gerät. Pro Messenger-Dienst werden häufig Tausende von Nachrichten unter einer Vielzahl von Kontakten und Gruppen ausgetauscht. Eine Nachricht kann dabei Text oder Mediendateien, wie Bilder, Videos oder Audios, enthalten. Die anfallenden Daten speichert jede Applikation in proprietären Formaten ab, welche meist sogar nach dem zugrundeliegenden Betriebssystem variieren. Dementsprechend bedarf es enormer Ressourcen, um ein mobiles Endgerät erfolgreich auszulesen, alle darauf befindlichen Messenger-Daten in eine auswertbare Form zu überführen und diese zu analysieren sowie in den Kontext mit anderen Daten oder Erkenntnissen aus den Ermittlungen zu setzen. Um diese Herausforderungen bewältigen zu können, existiert eine Vielzahl forensischer Programme, welche sich jedoch in ihrem funktionellen Schwerpunkt unterscheiden. Beispielsweise spezialisieren sich Cellebrite UFED[1], MSAB XRY[2] oder MOBILedit Forensic[3] auf die Extraktion von Daten. Auf der anderen Seite stellen Tools, wie MSAB XAMN[4], Nuix Investigate[5] oder der Mobile Network Analyzer (MoNA)[6], Funktionalitäten zur Visualisierung sowie zur effektiven Analyse von Massendaten bereit. Hierbei kommen unter anderem

[1] Website von Cellebrite UFED: https://cellebrite.com/de/cellebrite-ufed-de/, besucht am 13.07.2023
[2] Website von MSAB XRY: https://www.msab.com/de/product/xry-extract/, besucht am 13.07.2023
[3] Website von MOBILedit Forensic: https://www.mobiledit.com/mobiledit-forensic, besucht am 13.07.2023
[4] Website von MSAB XAMN: https://www.msab.com/de/product/analyze/, besucht am 13.07.2023
[5] Website von Nuix Investigate: https://www.nuix.com/technology/nuix-investigate, besucht am 13.07.2023
[6] MoNA-Website: https://forensic-science-investigation-lab.github.io/mona/, besucht am 13.07.2023

künstliche Intelligenz sowie weitere Data-Mining-Methoden zum Einsatz. Jedes Programm bietet dementsprechend eigene Arbeitsabläufe und Lösungen für spezifische Probleme. Jedoch stoßen Forensiker und Sachbearbeiter auf Schwierigkeiten, wenn sie die Vorteile und Ergebnisse der Programme kombinieren möchten, da jedes Programm eigene proprietäre Formate verwendet und ein Datenaustausch ohne Informationsverlust sowie ohne zusätzlichen Arbeitsaufwand meist nur zwischen Programmen des selben Herstellers möglich ist.

Um dieses Problem zu lösen, könnte in Zukunft Cyber-investigation Analysis Standard Expression (CASE) als Austauschformat für Tools und Organisationen verwendet werden. Hierbei handelt es sich um einen Ontologie-basierten Standard, welcher die Darstellung von Informationen aus Untersuchungen mit digitalen Beweismitteln ermöglicht [4]. CASE wird aktuell von einer internationalen Community entwickelt, welche aus einer Vielzahl von bedeutenden Organisationen, Institutionen und Unternehmen besteht, wie beispielsweise Europol, NIST, UNIL, Cellebrite, IBM, MSAB, Magnet Forensics, Nuix sowie Oxygen Forensics [5]. CASE stellt jedoch nicht nur lediglich ein Austauschformat dar. Der Ontologie-basierte Ansatz ermöglicht die Verknüpfung von Informationen und Ableitung von neuem Wissen [4]. Somit wird zum Beispiel eine Messenger- und Geräte-übergreifende Analyse ermöglicht. Darüber hinaus können komplexe Zusammenhänge in Verbindung mit anderen digitalen Spuren erkannt, Duplizierungen vermieden und die Aufrechterhaltung der Beweismittelkette sichergestellt werden [4].

1.2 Zielsetzung

Diese Arbeit untersucht, inwiefern eine Ontologie zur Repräsentation mobiler Kommunikation in der digitalen Forensik geeignet ist. Zusätzlich werden damit verbundene Herausforderungen und Chancen diskutiert. Als Grundlage für die Ontologie wird ein Teil von CASE verwendet, da dieser Standard von einem großen und namhaften Konsortium entwickelt wird und zukünftig in vielen forensischen Softwarelösungen implementiert werden soll [5]. Dabei untersucht diese Arbeit den aktuellen Entwicklungsstand und die Praxistauglichkeit von CASE. Insbesondere wird geprüft, inwieweit Informationen bezüglich mobiler Kommunikation im forensischen Kontext abgebildet werden können und ob Erweiterungen des Standards notwendig sind. Im experimentellen Teil dieser Arbeit erfolgt eine Abbildung des Datenmodells von dem Tool MoNA auf die entworfene Ontologie. MoNA ist eine forensische Analyse-Plattform zur effizienten Auswertung mobiler Kommunikation und verwendet semantische Technologien sowie künstliche Intelligenz [6]. Da aus den zugrundeliegenden Daten eine hohe Semantik durch MoNA abgeleitet werden kann, bietet sich eine Abbildung des MoNA-Modells auf die Ontologie an. Somit können die Ergebnisse von MoNA in einem standardisierten Format exportiert und anderen Tools zur Weiterverarbeitung bereitgestellt werden. Zum Testen der entworfenen Ontologie wird zudem ein SPARQL-Server aufgesetzt. Dieser Server ist für die Speicherung von Daten sowie für das Entgegennehmen und Beantworten von Anfragen verantwortlich.

1.3 Aufbau der Arbeit

Zu Beginn gibt Kapitel 2 eine kurze Einführung in die Mobilfunkforensik zur Einordnung dieser Arbeit. Anschließend diskutiert das Kapitel, wie generell von Daten auf Wissen geschlossen werden kann und wie sich Wissen formulieren lässt. Zudem wird zu Wissensrepräsentationen hingeführt und aufgezeigt, welche Anforderungen eine Wissensrepräsentation erfüllen muss. Daraufhin definiert das

Kapitel den Begriff der Ontologie und erklärt, was Ontologien charakterisiert. Im Anschluss folgt eine kompakte Einführung in die Sprachen zur Modellierung von Wissen und Ontologien, welche in CASE verwendet werden und deshalb für diese Arbeit von Relevanz sind. In Ergänzung dazu wird in die Abfragesprache SPARQL eingeführt. Grundlegende Fakten über die Unified Cyber Ontology (UCO) und CASE bilden den Abschluss des Kapitels. Kapitel 3 diskutiert relevante Literatur für diese Arbeit. Dabei wird insbesondere auf die Verwendung von Ontologien in der digitalen Forensik und zur Auswertung mobiler Kommunikation eingegangen. Darüber hinaus erfolgt eine Einordnung dieser Arbeit in den wissenschaftlichen Kontext. Anschließend leitet Kapitel 4 die Vorgehensweise zur Entwicklung der Ontologie her. Dabei wird ebenfalls das Datenmodell von MoNA und die Abbildung von diesem auf die Ontologie erklärt. Daraufhin führt das Kapitel aus, wie ein semantischer Webserver mittels Apache Tomcat und Apache Jena Fuseki aufgesetzt werden kann. Ebenso ist dokumentiert, welcher Anwendungstest bezüglich der entwickelten Ontologie über einen derartigen Server durchgeführt wurde. Kapitel 5 stellt die Ergebnisse dieser Arbeit dar. Hierbei wird die entwickelte Ontologie sowie deren Abbildung des MoNA-Datenmodells im Detail beschrieben und das Resultat des Anwendungstests aufgeführt. Zum Abschluss dieser Arbeit werden in Kapitel 6 die Ergebnisse zusammengefasst. Darüber hinaus diskutiert das Kapitel die Vorteile der Verwendung der entwickelten Ontologie und die Herausforderungen bei der Umsetzung in die Praxis. Dabei wird ebenfalls auf zukünftige Arbeiten eingegangen, welche an die erzielten Ergebnisse anknüpfen und diese erweitern können.

2 Grundlagen

2.1 Untersuchungen in der Mobilfunkforensik

Die Mobilfunkforensik befasst sich mit der Gewinnung und Auswertung digitaler Beweise von mobilen Geräten durch Anwendung von aus der IT-Forensik stammenden Techniken [7, S. 1–36]. Für die beiden beliebtesten Betriebssysteme für Smartphones, Android und iOS, existiert bereits umfangreiche Literatur bezüglich der forensischen Sicherung und Auswertung [8, 9]. Nach [10] treten bei der forensischen Untersuchung von Mobiltelefonen grundlegend folgende Daten auf:

- Speicherkartendaten von Telefonen
- SIM-Karteninformationen
- Nachrichten
- Standorte
- Daten aus den sozialen Netzwerken
- Anrufprotokolle
- Multimediadaten
- Kommunikationsnetzwerke

Diese Arbeit fokussiert sich hierbei unabhängig vom Betriebssystem auf Kommunikationsnetzwerke und Nachrichten sowie damit verbundene Standort- und Multimediadaten.

Eine forensische Untersuchung muss nach dem Leitfaden „IT-Forensik“ vom Bundesamt für Sicherheit in der Informationstechnik (BSI) spezifische Anforderungen an die Vorgehensweise erfüllen [11, S. 22 f.]. Unter anderem muss die angewandte Methodik von der Fachwelt beschrieben und akzeptiert sein, was dem Anforderungspunkt der *Akzeptanz* entspricht. Für neue Verfahren und Methoden sollte immer die Korrektheit nachgewiesen werden. Weiterhin müssen im Rahmen der *Glaubwürdigkeit* die Methoden robust sein und korrekt funktionieren, was ebenfalls nachweisbar sein sollte. Eine weitere wichtige Anforderung ist die *Wiederholbarkeit*, bei der Dritte mit derselben Vorgehensweise und dem gleichen Ausgangsmaterial immer dieselben Ergebnisse erzielen können sollten. Außerdem dürfen Spuren niemals unbemerkt im Rahmen der Untersuchung verändert werden. Die *Integrität* muss jederzeit sichergestellt und belegt sein. Des Weiteren müssen *Ursache und Auswirkungen* zwischen Ereignissen, Spuren und ggf. Personen nachvollziehbar sein. Zudem muss für jeden Schritt der forensischen Untersuchung eine angemessene *Dokumentation* erfolgen. Im Rahmen dieser Arbeit muss demzufolge beachtet werden, dass die Verwendung der zu entwickelnden Ontologie keine Verletzung der genannten Anforderungen des BSIs verursachen sollte.

In der IT-Forensik wird zwischen der Offline- und Online-Forensik unterschieden, welche in [11, S.13] folgendermaßen charakterisiert sind. Bei der Offline-Forensik, die auch Post-mortem-Analyse genannt wird, werden nichtflüchtige Spuren auf Datenträgerabbildern nach einem Vorfall untersucht. Insbesondere liegt hierbei ein Schwerpunkt auf der Identifikation sowie Auswertung gelöschter, versteckter und verschlüsselter Daten. Die Online-Forensik, auch als Live-Forensik bezeichnet, beschäftigt sich hingegen mit der Gewinnung und Analyse flüchtiger Daten. Darunter zählen zum Beispiel Daten aus dem Hauptspeicher oder Netzwerkverkehr. Die Verfahren der Online-Forensik werden hierbei bereits während des Vorfalls angewendet.

Für den Ablauf einer forensischen Untersuchung gibt das BSI im Leitfaden „IT-Forensik“ ein Prozessmodell an, an dem sich auch die Mobilfunkforensik orientiert [11, S. 60–65]. Dabei untergliedert das BSI den Prozess in die Abschnitte strategische Vorbereitung, operationale Vorbereitung, Datensammlung, Untersuchung, Datenanalyse und Dokumentation, wie in Abbildung 2.1 dargestellt.

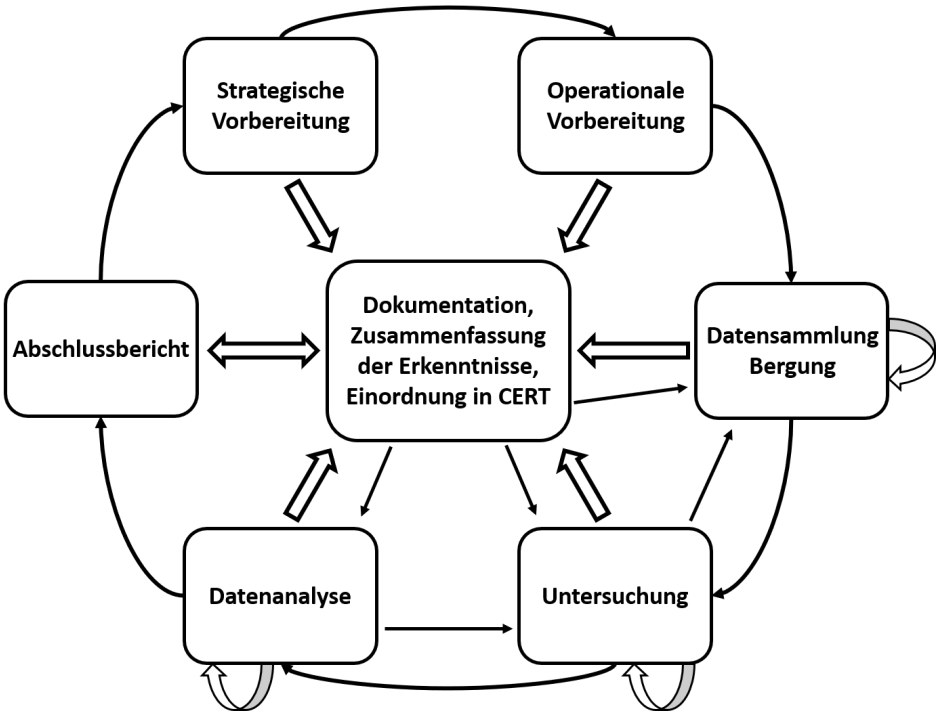

Abbildung 2.1: Die Untersuchungsabschnitte des forensischen Prozesses nach dem BSI. Adaptiert aus [11, S. 61].

Unter den strategischen Vorbereitungen versteht das BSI alle Maßnahmen, die proaktiv vor dem Eintreten eines Sicherheitsvorfalls oder einer Straftat ergriffen werden können [11, S. 60–65]. Dazu zählt beispielsweise die Aktivierung von Logdiensten. Operationale Vorbereitungen werden laut BSI nach Eintreten des Vorfalls beziehungsweise der Straftat durchgeführt. Hierunter werden alle Maßnahmen zusammengefasst, die vor der Datensammlung erfolgen, wie zum Beispiel die Identifikation und Enumeration potentieller Datenquellen. Danach schließt sich die Datensicherung an, bei der Abbilder von Massenspeichern erzeugt werden. Hierbei wird die Integrität der Beweismittel über kryptografische Verfahren abgesichert. Daraufhin werden in der Untersuchung forensisch wertvolle Daten aus den gesammelten Daten der Abbilder extrahiert. Im Abschnitt der Datenanalyse erfolgt nach dem BSI eine Detailanalyse dieser Daten. Dabei werden Verbindungen zwischen Daten hergestellt und gegebenenfalls die Urheberschaft ermittelt. Zeitliche Abläufe müssen nachvollziehbar und plausibel sein. In allen Untersuchungsabschnitten sollte laut BSI eine Dokumentation der durchgeführten Maßnahmen sowie die Zusammenfassung der Erkenntnisse erfolgen. Daraus leitet sich die prozessbegleitende Dokumentation ab. Aus ihr und der abschließenden Dokumentation, welche ein Gesamtbild aus den gesammelten Daten zeigt, wird im sechsten Untersuchungsabschnitt der Abschlussbericht erstellt. Aus diesem Bericht können nach dem BSI unter anderem wiederum Maßnahmen für die strategische Vorbereitung abgeleitet werden. Eine Ontologie zur Repräsentation mobiler Kommunikation, welche im Rahmen dieser Arbeit entwickelt werden soll, kann im Abschnitt der Datenanalyse Anwendung finden. Dazu müssen forensisch relevante Kommunikationsdaten von mobilen Endgeräten nach oder während der Extraktion in die Ontologie übertragen werden. Danach würde sich eine Vielzahl von Analyseoperationen mithilfe der Ontologie durchführen lassen.

2.2 Daten, Information und Wissen

Im Rahmen einer forensischen Untersuchung können eine Vielzahl von mobilen Endgeräten und die darauf befindlichen Kommunikationsdaten eine wichtige Rolle zur Aufklärung des Falles einnehmen. Die Analysten stehen nach erfolgter Sicherung der Geräte jedoch vor der Herausforderung, die erhaltenen Daten effektiv zu organisieren und auszuwerten. Bereits 1989 beschrieb Russell L. Ackoff in [12] die Zusammenhänge zwischen *Daten*, *Information*, *Wissen* sowie *Weisheit* in einer Hierarchie, welche Abbildung 2.2 darstellt. Die einzelnen Ebenen können durch das Verständnis der jeweils darunterliegenden Ebene erreicht werden. Daten definiert Ackoff als unverarbeitete Symbole, die aus Beobachtungen resultieren. Dabei repräsentieren sie Eigenschaften von Objekten, Ereignissen sowie ihrer Umwelt und können in jeder Form existieren. Daten bilden in der von Ackoff beschriebenen Hierarchie die unterste Schicht. Information, welche sich in der nächsthöheren Ebene befindet, wird aus Daten abgeleitet. Bei dieser Transformation werden die Daten für gewöhnlich reduziert und in eine nutzbare Form gebracht. Information ermöglicht eine Beschreibung der Daten und beantwortet darauf bezügliche Fragen, die mit den Wörtern *Wer*, *Was*, *Wann*, *Wo* und *Wie viele* beginnen. Die dritte Ebene bildet Wissen, das aus einer mit Semantik angereicherten Sammlung von nützlichen Informationen abgeleitet wird. An der Spitze der Hierarchie von Ackoff steht Weisheit, welche aus dem evaluierten Verständnis von Wissen erlangt werden kann. Während sich Daten, Information und Wissen auf die Vergangenheit beziehen, ermöglicht die Weisheit, fundierte Entscheidungen und Vorhersagen bezüglich der Zukunft zu treffen.

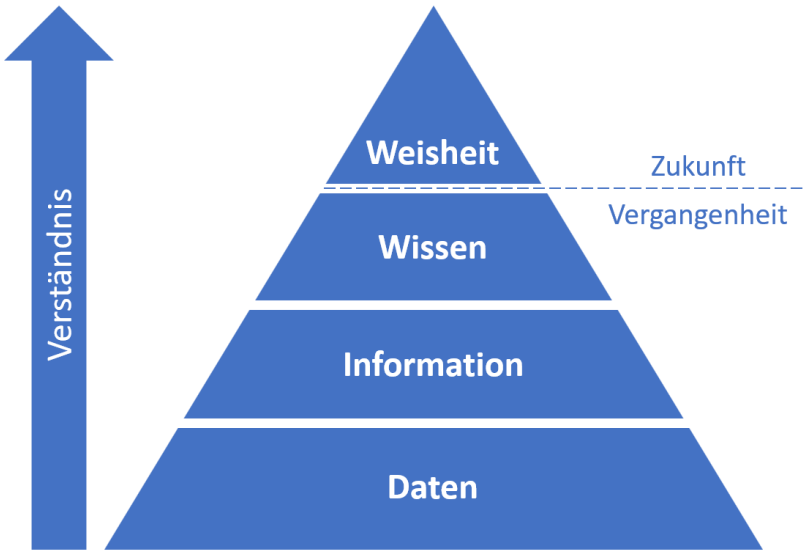

Abbildung 2.2: Die Hierarchie von Daten, Information, Wissen und Weisheit nach Russell L. Ackoff. Adaptiert aus [13].

Angewandt auf den Bereich der Strafverfolgung muss demzufolge mindestens die dritte Ebene (Wissen) erreicht werden, um Fragen in Bezug auf Straftaten klären zu können. Die höchste Ebene (Weisheit) ermöglicht darüber hinaus präventive Maßnahmen. Dadurch können Gefahren, zum Beispiel durch den effektiven Einsatz von Polizeiressourcen, verhindert werden.

Intelligente Systeme können die Organisation sowie Auswertung von Wissen erleichtern und somit zu einem tieferen Verständnis führen. Deshalb beschäftigt sich ein Forschungsfeld der künstlichen Intelligenz, die Wissensrepräsentation, damit, wie Wissen formuliert und repräsentiert werden kann,

um es anschließend maschinell verarbeiten zu können [14, S. 1–15]. Zur Formulierung von Wissen eignet sich hierfür Sprache. Sprache kann dabei als System von Symbolen zur Kommunikation verstanden werden [15]. Die Wissensrepräsentation ist nach [14, S. 1–4] ein Fachgebiet, welches sich mit der Verwendung formaler Symbole zur Darstellung von Glaubenssätzen beziehungsweise Wissen beschäftigt. Diese Definition wird zusätzlich damit ergänzt, dass eine unendliche Anzahl von Glaubenssätzen existiert, aber Symbole nur eine endliche Anzahl von ihnen repräsentieren können. Fehlendes Wissen kann durch eine Form des logischen Schlussfolgerns, welche Inferenz genannt wird, aus gegebenen Glaubenssätzen hergeleitet werden.

Die Bedeutung von Symbolen diskutierten bereits vor hundert Jahren Charles K. Ogden und Ivor A. Richards [16]. Dabei beschrieben sie ein Modell, welches als semiotisches Dreieck bekannt ist und die Zusammenhänge zwischen dem Symbol, dem dadurch hervorgerufenen Gedanken (Referenz) und dem realen Objekt (Referent) darstellt (siehe Abbildung 2.3). Nach den Autoren steht in der Sprache ein Symbol für ein reales Objekt wie Gegenstände, Sachverhalte, Ereignisse oder Ähnliches. Jedoch ruft ein Symbol für den Menschen nur ein gedachtes Bild hervor. Dieses Bild bezieht sich wiederum auf das Objekt aus der realen Welt. Der Bezug vom Symbol auf das reale Objekt erfolgt demzufolge nur mittelbar über den hervorgerufenen Gedanken. Die Bedeutung des Symbols hängt darüber hinaus vom Kontext der Kommunikation, der Absicht des Kommunizierenden und den Erfahrungen der Kommunikationsteilnehmer sowie deren gegenseitige Beziehung ab. Idealerweise sollten Symbol, Referenz und Referent eindeutig zusammengehören. Allerdings ist dies nicht immer gegeben. So könnte ein falsches Symbol für das Objekt verwendet werden, das Symbol einen irrtümlichen Gedanken hervorrufen oder das Objekt nicht korrekt wahrgenommen werden. In diesen Fällen entstehen nach Ogden und Richards Verwirrungen aufgrund von Missverständnissen.

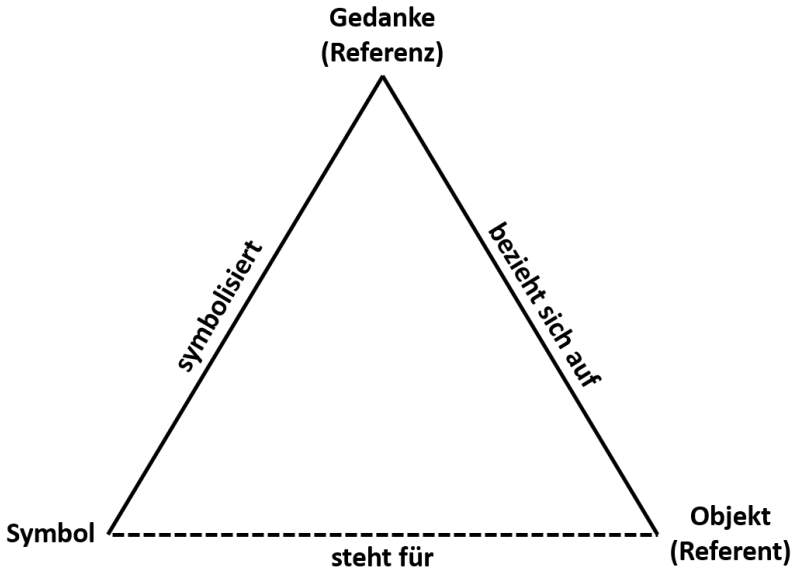

Abbildung 2.3: Das semiotische Dreieck nach C. K. Ogden und I. A. Richards, welches die Zusammenhänge zwischen Symbol, hervorgerufenen Gedanken und realem Objekt abbildet. Adaptiert aus [16].

Zur Vermeidung von Fehlinterpretationen müssen bestimmte Anforderungen erfüllt sein, damit durch Sprache formuliertes Wissen für Mensch und Maschine verständlich ist. Zur Wissensrepräsentation muss nach [14, S. 15 f.] eine formale Sprache folgende Aspekte besitzen:

- *Syntax*: Für eine Sprache muss spezifiziert sein, welche Gruppen von Zeichen in welcher Anordnung als korrekt gebildet gelten. Wohlgeformte Zeichenketten bilden die Sätze einer Sprache und drücken Aussagen aus.

- *Semantik*: Für eine Sprache muss spezifiziert sein, was die syntaktisch korrekt gebildeten Ausdrücke bedeuten sollen. Einige dieser Ausdrücke könnten nichts bedeuten. Deshalb muss klar sein, was mit einem Satz ausgedrückt wird.

- *Pragmatik*: Für eine Sprache muss spezifiziert sein, wie die bedeutungsvollen Ausdrücke in der Sprache verwendet werden sollen. Dabei ist immer der dazugehörige Kontext zu betrachten.

2.3 Ontologien zur Wissensrepräsentation

Ein Mittel zur Repräsentation von Wissen bieten Ontologien. Der Begriff *Ontologie* setzt sich aus den griechischen Wörtern *ontos* (deutsch: „sein“) und *logos* (deutsch: „Lehre“) zusammen und wird in der Philosophie als die „Lehre des Seins“ übersetzt [17].

2.3.1 Grundlegendes über Ontologien

Für Ontologien existieren eine Vielzahl verschiedener Definitionen. Thomas R. Gruber bezeichnet in [18] eine Ontologie als explizite Spezifikation einer Konzeptualisierung. Demzufolge beschreibt sie ein abstraktes Modell für eine bestimmte Domäne, wobei alle enthaltenen Konzepte genau definiert sind. Nach [17] stellt eine Ontologie einen Katalog, welcher Typen von Dingen aus einer Domäne enthält, bereit. Dabei liefert eine Sprache die Begriffe für Konzepte, Relationen und Prädikate zur Darstellung der Typen. Der Katalog ermöglicht die Erörterung von Themen in der Domäne. Damit bilden Ontologien einen fundamentalen Teil des Wissens über eine beliebige Domäne. Im Bereich der künstlichen Intelligenz werden dem Begriff der Ontologie nach [19] zwei sich ähnelnde Bedeutungen beigemessen. Zum einen wird sie als Repräsentationsvokabular verstanden, welches meist auf ein spezifisches Gebiet oder eine spezifische Domäne spezialisiert ist. Zum anderen existiert die Auffassung, dass eine Ontologie einem Wissensbestand entspricht, der einen bestimmten Bereich unter Verwendung eines Repräsentationsvokabulars beschreibt. Die bedeutsamsten Features von Ontologien sind Vokabular, Taxonomie, Inhaltstheorie sowie der Austausch und die Wiederverwendbarkeit von Wissen [17]. In den folgenden vier Absätzen werden die entsprechenden Ausführungen der Autoren zu den einzelnen Features kurz zusammengefasst.

Vokabular Mithilfe des Vokabulars können die Begriffe eines bestimmten Gebiets bezeichnet werden. Im Unterschied zu herkömmlichen, an den Menschen orientierten Vokabularien liefern Ontologien logische Aussagen, welche beschreiben, was die Begriffe bedeuten, wie sie miteinander in Beziehung stehen und wie sie in Beziehung gesetzt werden können oder nicht. Darüber hinaus definieren sie Regeln für die Kombination von Begriffen und ihre Beziehungen, wodurch Erweiterungen des Vokabulars realisiert werden. Eine Ontologie spezifiziert Begriffe mit eindeutigen Bedeutungen, wobei die Semantik unabhängig von dem Leser und Kontext ist. Sie ermöglicht maschinell verarbeitbares Allgemeinverständnis für Themen, welche die Begriffe bezeichnen. Die Bedeutungen der Begriffe in einer Ontologie können zwischen Benutzern und Anwendungen ausgetauscht werden. Eine Übersetzung der Begriffe in eine andere Sprache verändert dabei die Ontologie konzeptionell nicht.

Taxonomie Eine Taxonomie ist eine hierarchische Kategorisierung oder Klassifizierung von Entitäten innerhalb einer Domäne. Dabei werden Entitäten auf der Grundlage gemeinsamer ontologischer Merkmale gruppiert. Die Gruppierung erfolgt nach einem vorgegebenen System, wobei in einer guten Taxonomie die Entitäten in sich gegenseitig ausschließende, eindeutige Gruppen und Untergruppen unterteilt werden sollten. Jede Ontologie bietet eine Taxonomie in maschinenlesbarer und maschinell verarbeitbarer Form. Das Vokabular und die Taxonomie einer Ontologie bilden zusammen einen konzeptionellen Rahmen für die Diskussion, Analyse und Informationsbeschaffung in einer Domäne.

Inhaltstheorie Ontologien entsprechen Inhaltstheorien, da sie in einer Domäne existierende Klassen von Objekten, ihre Beziehungen sowie Begriffshierarchien identifizieren und spezifizieren. Somit stellen Ontologien Wissen aus einer Domäne auf eine strukturierte Weise dar. Dadurch ermöglichen sie verschiedene Arten der Konsistenzprüfung sowie die Verbesserung der Interoperabilität zwischen verschiedenen Anwendungen.

Austausch und Wiederverwendbarkeit von Wissen Der Hauptzweck von Ontologien ist es, Wissen durch Anwendungen zu teilen und wiederzuverwenden. Geteilte Ontologien ermöglichen den Aufbau spezifischer Wissensbasen, die spezifische Situationen beschreiben. Durch die Verwendung von Ontologien stützen sich die Wissensbasen auf eine gemeinsame Wissensstruktur und -organisation, was die Kompatibilität der von verschiedenen Anwendungen genutzten Domänenmodelle verbessert. In der Praxis ist ein Austausch und die Wiederverwendbarkeit von Wissen jedoch nicht in jedem Fall möglich. Denn beispielsweise werden verschiedene Sprachen zur Darstellung von Ontologien genutzt und nicht jedes Werkzeug zur Entwicklung von Wissensbasen unterstützt jede Sprache. Weiterhin existieren durch konkurrierende Ansätze und Arbeitsgruppen mehrere verschiedene Ontologien in denselben Domänen. Außerdem entwickelt sich Wissen über die Zeit, was zur Folge hat, dass auch die Ontologien stetig angepasst und erweitert werden müssen.

Nach [20] lässt sich eine Ontologie durch Klassen, Relationen und Instanzen repräsentieren. Klassen beschreiben dabei die Konzepte einer Domäne und werden durch Attribute charakterisiert, während Instanzen den Individuen einer Klasse entsprechen. Relationen sind spezielle Attribute, welche Beziehungen zwischen Klassen darstellen. Für Attribute und Relationen können Regeln definiert werden, um erlaubte beziehungsweise gültige Werte festzulegen.

2.3.2 Formulierung und Modellierung von Wissen

Resource Description Framework

Das Resource Description Framework (RDF) kann zur Formulierung von Wissen in einer Ontologie verwendet werden. So bildet es zum Beispiel die Basissprache für die CASE-Ontologie [21]. RDF wurde vom World Wide Web Consortium (W3C) konzipiert und ist in [22] bezüglich der XML Syntax dokumentiert. Diese Dokumentation dient hierbei als Grundlage der folgenden Beschreibung von Konzept und Syntax. Zur Repräsentation von Wissen werden in RDF eine Menge von Tripeln verwendet. Diese Menge wird als RDF-Graph bezeichnet. Ein Tripel besteht aus einem Subjekt, Prädikat und Objekt, wie in Abbildung 2.4 dargestellt. Subjekt sowie Objekt verweisen jeweils auf eine Ressource, die einer Sache aus der realen Welt entspricht. Ressourcen können beispielsweise physische Sachen, Dokumente, abstrakte Konzepte, Nummern oder Zeichenketten sein. Als Synonym dafür wird ebenfalls der Begriff „Entität“ verwendet. Das Prädikat stellt eine Eigenschaft dar

und verbindet das Subjekt mit dem Objekt. Ein Tripel stellt somit eine durch das Prädikat angegebene Beziehung zwischen den Ressourcen von Subjekt und Objekt dar. Ein Vokabular in RDF entspricht einer Sammlung von Internationalized Resource Identifiers (IRIs), die meist mit der gleichen Teilzeichenkette beginnen. Ein IRI dient zur weltweit eindeutigen Identifikation und Referenzierung einer Ressource, welche in diesem Kontext auch Referent genannt wird. Weiterführende Informationen zu IRIs sind in [23] gegeben. Subjekt, Prädikat und Objekt können jeweils einem IRI entsprechen. Alternativ kann ein Objekt stattdessen ein Literal sein, die zur Beschreibung von Werten wie Zeichenketten, Nummern oder Datumsangaben genutzt werden. Darüber hinaus kann für ein Subjekt oder Objekt anstelle einer IRI oder einem Literal ein leerer Knoten („Blank Nodes“) vorkommen. Dieser Knoten verweist auf die Existenz eines Individuums mit spezifischen Attributen. Jedoch ist hierbei keine Identifikation und externe Referenzierung möglich.

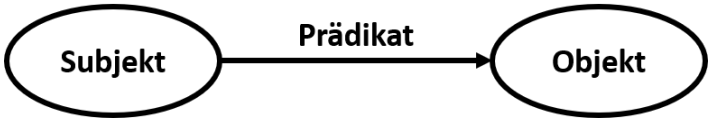

Abbildung 2.4: Die drei Komponenten eines RDF-Tripels: Subjekt, Prädikat, Objekt. Adaptiert aus [22].

Turtle

Eine vom W3C entwickelte und in [24] dokumentierte Erweiterung von RDF ist Turtle. Hierbei erleichtert sie insbesondere die Lesbarkeit von RDF-Tripeln und führt weitere Datenstrukturen ein. Unter anderem ermöglicht Turtle die Abkürzung von IRIs durch die Einführung von Präfixen, welche jeweils einen sogenannten Namensraum (Namespace) bilden. Darüber hinaus können über ein Semikolon Tripel mit identischem Subjekt verbunden werden. In diesem Fall wird das Subjekt nur im ersten Tripel ausgeschrieben und in den folgenden weggelassen. Besitzen Tripel dasselbe Subjekt und Prädikat, so dürfen diese per Komma verknüpft werden. Dabei entfallen Subjekt und Prädikat in den entsprechenden Tripeln, welche nach dem ersten Tripel folgen. Weiterhin kann in Turtle der Datentyp von Literalen durch eine IRI eindeutig angegeben werden. Hierzu folgen nach dem Wert des jeweiligen Literals die Zeichen `^^` und daraufhin die IRI für den entsprechenden Datentyp. Weitere Features von Turtle, wie beispielsweise „Collections“ oder verschachtelte, anonyme Blank Nodes, können in der bereits erwähnten Dokumentation nachgelesen werden, spielen jedoch für das grundlegende Verständnis dieser Arbeit keine Rolle.

RDF Schema

Neben Turtle findet auch das RDF Schema (RDFS), welches ebenfalls vom W3C entwickelt wird, in CASE Anwendung [21]. RDFS ist in [25] dokumentiert und stellt ein Vokabular für RDF-Daten zur Modellierung bereit. Alles in einem RDF-Modell ist eine Ressource (`rdfs:Resource`). Durch die Verwendung spezifischer Prädikate kann verstärkt Semantik ausgedrückt werden. Unter anderem erlaubt RDFS die Definition von Klassen über das Prädikat `rdfs:Class` sowie die Instanziierung von Klassen über das Prädikat `rdf:type`. Prädikate können als `rdf:Property` definiert werden. Außerdem ermöglicht RDFS Beschränkungen bezüglich des Definitions- und Wertebereichs von Subjekten über die Prädikate `rdfs:domain` und `rdfs:range`. Weiterhin können hierarchische Beziehungen abgebildet werden. Das Prädikat `rdfs:subClassOf` wird zur Definition von Sub- und Superklassen verwendet. Unter- oder übergeordnete Eigenschaften können über das Prädikat `subPropertyOf`

definiert werden. Eine Auswahl weiterer erwähnenswerter RDFS-Prädikate sind in Tabelle 2.1 aufgeführt. Da RDFS auf einer formalen Semantik basiert, ermöglicht es das Ziehen von validen und logischen Inferenzen.

Tabelle 2.1: Auswahl und Erläuterung einiger erwähnenswerter RDFS-Prädikate.

RDFS-Prädikat	Erläuterung zur Verwendung
`rdfs:comment`	Zum Hinzufügen von Kommentaren, gewöhnlich in Textform
`rdfs:label`	Zur Definition eines lesbaren Namens für eine Ressource
`rdfs:seeAlso`	Zur Referenzierung auf eine Ressource, welche das Subjekt erklärt
`rdfs:isDefinedBy`	Zur Referenzierung auf eine Ressource, welche das Subjekt definiert; Subeigenschaft von `rdfs:seeAlso`

OWL 2 Web Ontology Language

Zur Definition von Klassen, Eigenschaften und Beziehungen nutzt CASE zudem das Vokabular von der OWL 2 Web Ontology Language (OWL 2) [21]. Die Sprache OWL 2 ist eine Überarbeitung sowie Erweiterung von OWL Web Ontology Language (OWL) und in [26] dokumentiert. Beide Versionen wurden vom W3C entwickelt und können mit RDF sowie RDFS kombiniert werden. Der Austausch von OWL 2-Ontologien erfolgt dabei hauptsächlich als RDF-Dokumente. Die folgenden Ausführungen stammen aus der Dokumentation von OWL [27], gelten aber ebenfalls für OWL 2.

Das Vokabular von OWL kann über die IRI `http://www.w3.org/2002/07/owl#` als Präfix angegeben werden und wird in der Regel mit `owl` abgekürzt. Für OWL existiert eine Turtle Syntax. Dabei setzen sich OWL-Axiome aus drei Bestandteilen zusammen. Analog zu RDFS existieren in OWL Klassen (*Classes*) und Eigenschaften (*Properties*). Zudem entsprechen Individuen (*Individuals*) in OWL den Klasseninstanzen in RDFS. OWL besitzt zwei vordefinierte Klassen. `owl:Thing` ist die Superklasse aller Individuen. `owl:Nothing` stellt eine leere Klasse dar. Individuen werden über die Zugehörigkeit zu einer Klasse durch das Prädikat `a` definiert. Alternativ über die Definition als `owl:NamedIndividual` benötigen Individuen jedoch keine Klassenzugehörigkeit. In OWL existieren zwei verschiedene Arten von Eigenschaften. Objekteigenschaften (`owl:ObjectProperty`) haben Klassen als Wertebereich. Wiederum werden Datentypen zur Definition des Wertebereichs von Datentypeigenschaften (`owl:DatatypeProperty`) genutzt. Um Klassenhierarchien darzustellen, kann das RDFS-Prädikat `rdfs:subClassOf` wiederverwendet werden. OWL stellt zudem Prädikate zum Ausdruck von Gleichheit und Ungleichheit bereit. Identische Klassen können über `owl:equivalentClass` definiert werden, identische Individuen mittels `owl:sameAs`. Auf der anderen Seite wird `owl:disjointWith` für sich unterscheidende Klassen und `owl:differentFrom` für nicht identische Individuen verwendet. Darüber hinaus ermöglicht OWL durch Nominale und logische Konstruktoren die Definition komplexerer Klassen. „Property Restrictions“, welche entweder die Werte oder Kardinalität beschränken können, dienen als Eigenschaften zur Beschreibung dieser Klassen. Des Weiteren erlaubt OWL die Definition von Beziehungen zwischen Eigenschaften. Weiterführende Informationen zur Verwendung des OWL-Vokabulars können aus der bereits erwähnten Dokumentation entnommen werden.

2.3.3 Erstellung einer Ontologie

Das Konsortium von CASE empfiehlt den Leitfaden von Noy und McGuinness [20] zum Verständnis und zur Adaption der CASE-Ontologie [4]. Deshalb werden in diesem Abschnitt die notwendigen Schritte zur Erstellung einer Ontologie aus diesem Leitfaden zusammengefasst. Die Abfolge der Schritte stellt Abbildung 2.5 als Prozesskette dar.

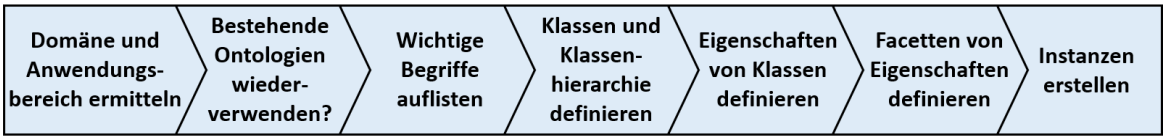

Abbildung 2.5: Prozesskette zur Erstellung einer Ontologie nach [20].

1. Ermitteln der Domäne und des Anwendungsbereichs

Zuerst wird die Domäne und der genaue Anwendungsbereich der Ontologie definiert. Dazu ist die Beantwortung folgender Fragen notwendig:

- Welche Domäne soll die Ontologie abdecken?
- Wofür soll die Ontologie verwendet werden?
- Für welche Art von Fragen sollen die Informationen in der Ontologie Antworten liefern?
- Wer wird die Ontologie verwenden und pflegen?

Durch Beantwortung dieser Fragen lässt sich der Anwendungsbereich der Ontologie einschränken und die Anforderungen einfacher identifizieren.

2. Erwägung der Wiederverwendung bestehender Ontologien

Als nächstes erfolgt die Recherche, ob bereits eine Ontologie existiert, welche die ermittelte Domäne abdeckt. Wenn dies der Fall ist, wird in Erwägung gezogen, ob die Ontologie wiederverwendet werden sollte. Dabei wäre ein Vorteil, dass die Interaktion mit anderen Tools und Systemen, welche die Ontologie bereits verwenden, ermöglicht wird.

3. Auflistung wichtiger Begriffe in der Ontologie

Im dritten Schritt werden alle Begriffe aufgelistet, über die Aussagen getroffen oder die dem Anwender erklärt werden sollen. Darunter fallen zusätzlich alle Begriffe, die über- oder untergeordnet zu anderen Begriffen sind oder sich zu deren Beschreibung verwenden lassen. Dabei sollte die Liste vor allem möglichst umfangreich sein, wobei in diesem Schritt Überschneidungen in Kauf genommen werden.

4. Definition der Klassen und Klassenhierarchie

Daraufhin folgt die Definition der Klassen und Klassenhierarchie unter Verwendung der erfolgten Auflistung aus Schritt 3. Die Klassenhierarchie kann dabei *top-down*, *bottom-up* oder in einer Kombination davon erfolgen. Im *top-down*-Prozess werden zuerst die allgemeinsten Konzepte der Domäne definiert und danach Spezifikationen hinzugefügt, die sich den Konzepten in der Hierarchie immer mehr unterordnen. Der *top-down*-Ansatz startet mit den spezifischsten Klassen und gruppiert diese Klassen anschließend in der Hierarchie steigend nach allgemeineren Konzepten. Bei der Kombination von *bottom-up* und *top-down* werden zuerst die wichtigsten Begriffe definiert. Danach schließt sich die Verallgemeinerung in der Hierarchie nach oben und Spezialisierung in der Hierarchie nach unten an. Im vierten Schritt wird insbesondere auf die Vermeidung von Überschneidungen, Widersprüchen und Zyklen geachtet.

5. Definition der Eigenschaften von Klassen

Als Nächstes folgt die Beschreibung der inneren Struktur der Klassen anhand von Eigenschaften. Nach Durchführung von Schritt 4 sollten in der Auflistung aus Schritt 3 Begriffe übrig sein, welche zur Beschreibung der definierten Klassen geeignet sind. Diese Begriffe werden den jeweiligen Klassen als deren Attribute zugeordnet. Die Zuordnung sollte dabei je an die möglichst allgemeinste Klasse erfolgen. Alle Subklassen erhalten in diesem Fall das entsprechende Attribut durch Vererbung.

6. Definition der Facetten von Eigenschaften

Die definierten Eigenschaften können verschiedene Facetten haben, welche den Wertetyp, erlaubte Werte, die Anzahl der Werte sowie andere Merkmale der Werte beschreiben. Die Kardinalität definiert dabei, wie viele Werte eine Eigenschaft haben kann. Darüber lässt sich die Kardinalität durch Angabe eines Minimums und eines Maximums genauer spezifizieren. Der Wertetyp ist in den meisten Fällen ein String, eine Nummer, ein Boolean, eine Enumeration oder ein Typ einer Instanz. Zudem können der Definitionsbereich (*domain*) sowie der Wertebereich (*range*) für jedes Attribut definiert werden. Der Wertebereich gibt an, welche Klassen für Attribute vom Typ Instanz erlaubt sind. Währenddessen umfasst der Defintionsbereich die Klassen, denen ein Attribut zugeordnet ist.

7. Erstellung von Instanzen

Im letzten Schritt werden die Individuen der Klassen in der Hierarchie erstellt. Dazu wird zuerst die geeignete Klasse gewählt, eine Instanz der Klasse erstellt und abschließend die Attribute dieser Instanz mit Werten gefüllt.

2.3.4 SPARQL zur Abfrage von Wissen

Zur Realisierung von Anfragen für RDF-Modelle kann die vom W3C entwickelte Sprache SPARQL verwendet werden. Die zugehörige Syntax und Semantik ist in [28] definiert. Anfragen in SPARQL sind über verschiedene Datenquellen ausführbar, solange die Daten im RDF-Format gespeichert oder dargestellt werden. Variablen sind in SPARQL an die RDF-Begriffe gebunden und lassen sich über ein `SELECT`-Statement, ähnlich wie in SQL, abfragen. SPARQL basiert auf der Turtle-Serialisierung und dem Abgleich von Graph-Mustern. Ein Graph-Muster entspricht hierbei einem RDF-Tripel, das an jeder beliebigen Stelle (Subjekt, Prädikat, Objekt) Variablen enthalten kann. Darüber hinaus können Graph-Muster kombiniert werden, um komplexere Anfragen zu bilden. Des Weiteren unterstützt SPARQL unter anderem Aggregation, Unterabfragen, Negation und Beschränkungen durch Filter. Als Ergebnis einer `SELECT`-Anfrage werden alle Variablen, die dem Anfrage-Muster entsprechen, in einer Tabelle zurückgegeben. Die entsprechende Ausgabe kann über ein lokales Application Programming Interface (API) verarbeitet oder in JSON, XML, CSV oder TSV serialisiert werden. Zusätzlich existieren neben `SELECT` die Befehle `ASK`, `DESCRIBE` und `CONSTRUCT`. Ein `ASK`-Statement prüft, ob mindestens ein Ergebnis für die Anfrage existiert und gibt im XML- oder JSON-Format entweder `true` oder `false` zurück. `DESCRIBE` liefert einen RDF-Graphen mit Daten über die Ressourcen im RDF/XML- oder Turtle-Format. `CONSTRUCT`-Statements erzeugen jeweils einen RDF-Graphen im RDF/XML- oder Turtle-Format nach einer Vorlage, die dem Graph-Muster der Anfrage entspricht und bei der die Variablen ersetzt werden.

2.4 Unified Cyber Ontology

Die UCO ist eine von einer Community entwickelte Ontologie zur standardisierten Informationsdarstellung in der gesamten Domäne der Cybersicherheit [29]. Laut eigenen Angaben können basierend auf dieser Ontologie spezifische Informationsdarstellungen erstellt werden, welche sich auf einzelne Teilbereiche fokussieren. Dazu eignet sich die Verwendung von passenden Teilmengen der UCO. Dadurch lassen sich gemeinsame APIs nutzen und Informationen aus verschiedenen Teilbereichen übergreifend verwenden. Die Community führt folgende Anwendungsfälle für die UCO auf [29]:

- Standardisierter Austausch großer und vielfältiger Mengen von Cyberinformationen unter Vermeidung von Duplikaten
- Interoperabilität zwischen Tools und Systemen, welche Automatisierung, Normalisierung, Kombination und Korrelation ermöglichen
- Kombination von Informationen aus verschiedenen Teilbereichen der Cyberdomäne
- Bereitstellung von Strukturen zur Verbesserung der intelligenten Analyse
- Verbesserung des Testens von Tools und der Validierung ihrer Ergebnisse
- Zugangskontrolle zu privilegierten, geschützten und persönlichen Informationen
- Unterstützung für benutzerdefinierte oder nicht standardisierte Strukturen

Die Subdomänen der UCO umfassen die digitale Untersuchung, Cyberrisikomanagement, Cyber Threat Intelligence und IT-Sicherheit für die Lieferkette [30]. CASE bildet hierbei die Subdomäne für digitale Untersuchungen und wird nach eigenen Angaben von der „CASE Opensource Community“ verwaltet. Für weitere Subdomänen sollen eigene Communities gebildet werden.

Zum Veröffentlichungszeitpunkt dieser Arbeit befindet sich die UCO in der Version `1.2.0`[7] vom 29.03.2023. Unter `https://ontology.unifiedcyberontology.org/` kann die Dokumentation der jeweils aktuellsten Version aufgerufen werden (Stand: 28.09.2023). Die Ontologie ist öffentlich auf GitHub[8] verfügbar.

2.5 Cyber-investigation Analysis Standard Expression

CASE ist eine Erweiterung der UCO, welche die Domäne von digitalen Untersuchungen im Kontext von Kriminalität, Unternehmen und Geheimdiensten repräsentiert [31]. CASE unterstützt das Verknüpfen von Daten und bietet neue Möglichkeiten für kontextbezogene Analysen, Mustererkennungen, maschinelles Lernen sowie Visualisierungen. Darüber hinaus liefert CASE eine standardisierte Darstellung der Beweismittelkette, wodurch sie im ganzen Zyklus von Cyberuntersuchungen aufrechterhalten wird und nachgewiesen werden kann [31]. CASE wird seit 2014 entwickelt und entsprang aus der Zusammenarbeit zwischen DoD Cyber Crime Center und MITRE unter der Führung von Dr. Eoghan Casey und Sean Barnum, wobei auch das National Institute of Standards and Technology einbezogen wurde [32]. Aufgrund des internationalen Interesses wurde die Initiative zu einem sich entwickelnden Open-Source-Standard [32]. Mittlerweile besteht die CASE-Community aus einem breit aufgestellten und namhaften Konsortium von Forschungsinstituten, staatlichen Einrichtungen sowie Unternehmen aus dem Bereich der forensischen Softwareentwicklung [5].

In CASE dienen `ObservableObjects` als Klasse zur Repräsentation von Individuen und werden durch zugehörige `Facets` charakterisiert [32]. `ObservableObjects` sind beispielsweise Datenquellen und digitale Objekte wie Dateien und Ordner sowie Nachrichten, Dokumente, Multimedia und Logs. Die Instanzen von *Facets* hingegen sind keine domänenspezifischen Individuen. `Facets` repräsentieren die Eigenschaften der zugeordneten `ObservableObjects`. Ein `ObservableObject` kann dabei ein oder mehrere *Facets* besitzen, wobei die Kombination von mehreren `Facets` als *Duck-Typing* bezeichnet wird [32]. Bei diesem Konzept werden die `ObservableObject` nicht über ihren Typ sondern über ihre Eigenschaften, also über die zugehörigen `Facets`, beschrieben [33]. Abbildung 2.6 zeigt ein schematisches Beispiel für ein `ObservableObject`, das anhand von drei `Facets` charakterisiert wird. Das obere `Facet` stellt mit einem Dateinamen und der Änderungszeit Eigenschaften zur Beschreibung von Dateien bereit. Das mittige `Facet` spezifiziert das `ObservableObject` als Rastergrafik durch die beiden Eigenschaften Bildtyp und Methode der Bildkompressionsmethode. Das untere `Facet` ist benutzerdefiniert und dient zur Annotation der Bilder von Waffen.

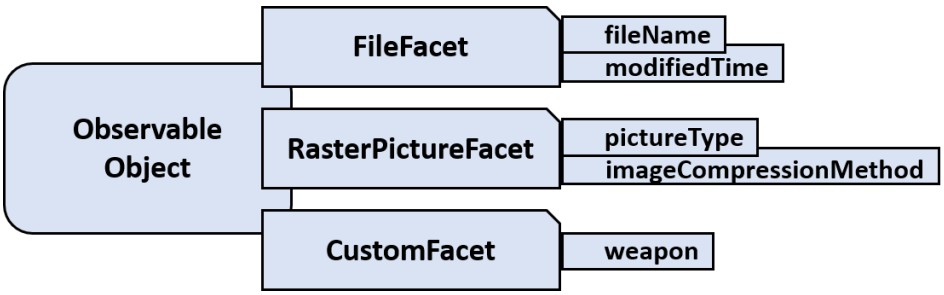

Abbildung 2.6: Beispiel für ein `ObservableObject` in CASE, welches durch drei `Facets` mit spezifischen Eigenschaften charakterisiert wird und somit Rastergrafiken von Waffen repräsentiert. Adaptiert aus [32].

[7] UCO `1.2.0` Release Notes: `https://unifiedcyberontology.org/releases/1.2.0/`, besucht am 28.09.2023
[8] UCO auf GitHub: `https://github.com/ucoProject/UCO`, besucht am 28.09.2023

Zum Veröffentlichungszeitpunkt dieser Arbeit befindet sich die CASE-Ontologie in der Version `1.2.0`[9] vom 29.03.2023. Unter `https://ontology.caseontology.org/` kann die Dokumentation der jeweils aktuellsten Version aufgerufen werden (Stand: 28.09.2023). Die Ontologie ist öffentlich auf GitHub[10] verfügbar. Die folgenden Kapitel gehen unter anderem auf weitere Details bezüglich des Designs und der Spezifikation der CASE-Ontologie ein. Insbesondere wird in Kapitel 4 und Kapitel 5 ausführlich die Verwendung eines Teils der Ontologie zur Repräsentation mobiler Kommunikation behandelt und diskutiert.

[9] CASE `1.2.0` Release Notes: `https://caseontology.org/releases/1.2.0/`, besucht am 28.09.2023
[10] CASE auf GitHub: `https://github.com/casework/CASE`, besucht am 28.09.2023

3 Verwandte Arbeiten

Im Jahr 2010 prognostizierte Simson L. Garfinkel in [34], dass sich das goldene Zeitalter der Computerforensik dem Ende nähere. Um eine aufkommende Krise zu überstehen, sieht der Autor die Notwendigkeit, die Forschung in dem Gebiet der digitalen Forensik effizienter zu gestalten. So fehlen bisher standardisierte Abstraktionen und Datenformate, weshalb Forscher mehr Zeit für die Beschaffung und Aufbereitung von Daten benötigen. Zudem wird die Entwicklung von Systemen immer aufwendiger und Forschungsprodukte können schwieriger miteinander verglichen werden. Das größte Problem, das der Autor feststellt, ist das Fehlen von Austauschformaten. Diese Formate würden die Entwicklung von interoperablen Forensik-Tools ermöglichen. Auch über einem Jahrzehnt nach der Veröffentlichung der Arbeit existieren diese sowie weitere Herausforderungen und Limitierungen in den Teilbereichen der digitalen Forensik, welche in [35] folgendermaßen aufgeführt werden:

- Fehlen eines standardisierten Untersuchungsansatzes
- Fehlen eines harmonisierten Ansatzes zur Sammlung von Beweisen
- Teilbereich übergreifende Zusammenhänge
- Fehlen eines standardisierten Datenformats
- Wachsende Komplexität von Geräten
- Heterogenität von Datenquellen

Hierbei stellt eine domänenbasierte Ontologie einen Lösungsansatz dar [35]. Sie könnte als standardisierte Basis dienen und die Abhängigkeiten zwischen den verschieden Teilbereichen aufzeigen. Ebenso betonen Allyson M. Hoss und Doris L. Carver in [36], dass Ontologien zur einheitlichen und formalen Repräsentationen von Wissen aus dem Bereich der digitalen Forensik genutzt werden können. Somit lasse sich unter anderem die Inkompatibilität zwischen forensischen Tools beheben und eine automatisierte Analyse realisieren. Außerdem könnten forensische Experten zur Untersuchung verwendete Tools besser analysieren und deren Korrektheit überprüfen. Beide Quellen beschreiben jedoch nur erste Ansätze und keine konkrete Ontologie, die sich als Standard eignen würde.

In der Vergangenheit wurden bereits einige domänenspezifische XML-Sprachen in verschiedenen Bereichen der digitalen Forensik definiert, um ein Austauschformat für forensische Informationen zu schaffen. Beispielsweise verwendet Digital Forensics XML (DFXML) ein XML-Vokabular, welches zur Beschreibung des Inhalts von Disk-Images verwendet werden kann [37]. Ein weiteres Beispiel ist XLIVE, das ein XML-basiertes Framework zur Sammlung von Daten nutzt, welche bei Untersuchungen eines Windows-Systems im Bereich der Online-Forensik auftreten [38]. Darüber hinaus existieren generische Ansätze wie Digital Evidence Exchange (DEX) [39] und XIRAF [40], welche ebenfalls über ein XML-Format den Datenaustausch zwischen forensischen Tools realisieren. Die in diesem Absatz aufgeführten XML-Formate ermöglichen zwar den Austausch von forensischen Informationen, jedoch kann kaum die Semantik von Objekten und Attributen sowie ihre Beziehungen untereinander ausgedrückt werden. Deshalb ergeben sich starke Einschränkungen beim Ziehen von logischen Schlussfolgerungen aus dem gegebenen Datensatz.

Andere Arbeiten nutzen RDF für ihre Datenmodelle. So findet es in dem forensischen Format AFF4 Anwendung, welches als eine Art Container für digitale Beweise dient [41]. RDF ermöglicht in diesem Fall die Erstellung ausdrucksstarker Attribute mit standardmäßigen oder proprietären Typen, wodurch

Metadaten über die erhaltenen Beweise festgelegt werden können. Ebenso lassen sich in AFF4 Metadaten über den Untersuchungsprozess und Informationen, die während der Untersuchung generiert wurden, anhand von RDF-Statements ausdrücken [42]. Somit kann die Beweismittelkette gestärkt werden.

Ein Review betrachtet in der Zeitspanne von 2009 bis 2022 ausführlich die wissenschaftliche Literatur über Ontologien in der digitalen Forensik [43]. Dabei fassen die Autoren zusammen, dass in diesem Bereich zwar eine Vielzahl von Ansätzen und Entwürfen von Ontologien existiere, aber bisher keine Ontologie allgemein akzeptiert sei, da verschiedenste Anforderungen bestünden. Eine Hauptaufgabe sei die automatisierte Erstellung strukturierter Daten aus verschiedenen Datenquellen. Eine Ontologie soll hierbei zur Beschreibung von Daten aus heterogenen Quellen dienen. In den Ergebnissen und in der Diskussion führen die Autoren ebenfalls CASE auf. Dabei betonen sie, dass im Vergleich zu anderen Ontologien mit CASE eine solide Grundlage zur Formalisierung geschaffen worden sei. An vorherigen Ansätzen kritisieren die Autoren, dass durch diese zwar generelles, aber kaum forensisches Wissen abgebildet werden kann. CASE selbst bedarf laut den Autoren noch technischer Annotationen oder der Modellierung der Akquise. Durch eine Erweiterung mit Regeln der Semantic Web Rule Language könnte darüber hinaus das logische Schlussfolgern ermöglicht werden. Nach der Veröffentlichung der Arbeit erschienen der Website[11] von CASE zufolge zwei neue Versionen. In diesem Zusammenhang ist eine Überprüfung sinnvoll, inwieweit die Forderungen der Autoren bereits umgesetzt wurden.

Dosis et al. stellen in [44] einen Ansatz vor, digitale Beweise ontologisch zu präsentieren und integrieren. Somit wollen die Autoren bestehende Analysetechniken aus der Forensik erweitern und insbesondere die (Teil-)Automatisierung von Untersuchungsprozessen ermöglichen. Dazu stellen sie eine semantisch gestützte Methode vor, deren Struktur Abbildung 3.1 darstellt. Der erste Schritt ist das Sammeln von Daten durch forensische Akquise-Techniken. Danach folgt die Transformation der meist heterogenen Daten in semantische Repräsentationen, wobei in dem beschriebenen Ansatz Ontologien unter Nutzung von OWL verwendet werden. Im nächsten Schritt wird die Ontologie an einen semantischen OWL-Reasoner übergeben, welcher Schlussfolgerungen aus der Datenquelle und Ontologie zieht. Beispielweise lassen sich somit Klassenzugehörigkeiten ableiten und Eigenschaften feststellen. Zudem wird ein regelbasierter Reasoner verwendet, um anhand komplexerer, fallspezifischer Regeln zusätzliche Aussagen zu schlussfolgern. Das ontologische und regelbasierte Schlussfolgern sollte dabei laut den Autoren iterativ durchgeführt werden. Regelbasierte Reasoner können somit ontologische Schlussfolgerungen berücksichtigen und wiederum für den ontologischen Reasoner neue Verknüpfungen zwischen Individuen liefern. Im letzten Schritt dieser Methode navigieren die Ermittler über die resultierende Wissensbasis. Zur Formulierung und Ausführung von Anfragen wird hierzu SPARQL verwendet. Letztendlich können Ermittler somit fallrelevante Ereignisse und Spuren identifizieren und abrufen. Für die praktische Umsetzung der aufgeführten Methode haben die Autoren mehrere, domänenspezifische Ontologien entwickelt. Jedoch sind diese Ontologien nicht mehr unter dem in der Quelle angegebenen Link abrufbar. Die Autoren beschrieben in ihrer Arbeit zudem nur zwei der entworfenen Ontologien, eine Ontologie für die Medienspeicherung und eine Ontologie für die Netzwerkverkehr-Analyse. Eine semantische Repräsentation von mobiler Kommunikation wurde dabei nicht behandelt.

[11] CASE Releases: https://caseontology.org/releases/archive/, besucht am 28.09.2023

Abbildung 3.1: Abfolge einer semantisch gestützten Methode zur Repräsentation und Integration digitaler Beweise. Adaptiert aus [44].

Schatz et al. beschreiben in [45] den prototypischen Entwurf einer Ontologie zur Repräsentation von forensisch relevanten Events, wobei sie OWL nutzten. Mit einem Proof of Concept zeigten die Autoren, dass unter Verwendung der Ontologie und einer Sprache für Korrelationsregeln Szenarien aus der Computerforensik anhand gegebener Event-Logdaten automatisch detektiert werden können. In [46] erweiterten dieselben Autoren ihren Ansatz. Dabei ermöglichten sie insbesondere das Formulieren von Korrelationsregeln, welche Entitäten und Events aus unterschiedlichen Domänen kombinieren. Da sich die beiden Arbeiten jedoch auf eine benutzerdefinierte und begrenzte Regelsprache stützen, ist der geschilderte Ansatz bezüglich der Erweiterbarkeit und Übernahme in die Praxis eingeschränkt.

Die Digital Investigation Ontology (DIALOG) ist ein auf einer Ontologie basierendes Framework zum Verwalten, Wiederverwenden und Analysieren von Wissen aus digitalen Untersuchungen [47]. Dabei stellt es ein allgemeines, von Anwendungen unabhängiges Vokabular zur Verfügung, über das eine Untersuchung in verschiedenen Detailgraden beschrieben werden kann. Hauptsächlich lässt sich DIALOG in vier grundlegende Subontologien unterteilen: Typen der Untersuchung, Typen der Datenspeicherorte, Typ der Daten und Tools zum Finden der Daten. Zur Veranschaulichung anhand eines Anwendungsbeispiel beschränken sich die Autoren jedoch nur auf die semantische Repräsentation der Windows Registry. Dabei lassen sich Schlussfolgerungen mithilfe von Regeln ziehen. In zukünftigen Arbeiten muss die Ontologie allerdings noch erweitert werden, um die Domäne der digitalen Forensik breiter und detaillierter repräsentieren zu können.

Ellison et al. konzipierten in [48] eine Ontologie zur Repräsentation reaktiver Techniken der digitalen Forensik, welche nach Funktionen klassifiziert sind. Darüber hinaus enthält die Ontologie zusätzliche Informationen, sodass sie in verwandte Systeme zusammengeführt werden kann. Darin können die Beziehungen zwischen den Techniken und anderen Facetten des digitalen Ermittlungsprozesses definiert werden. Die Autoren orientierten sich beim Entwurf der Ontologie dabei an den drei Prozessphasen des ISO-Standards 27043 [49]. Daraus folgend können Techniken aus den Phasen der Initialisierung, Akquise und Untersuchung repräsentiert werden. Ellison et al. sehen dabei die Nützlichkeit der Ontologie darin, eine digital forensische Bibliothek erstellen zu können. Somit könnte sie als international anwendbares Framework für Metadaten zum Klassifizieren von reaktiven Techniken der digitalen Forensik dienen. Jedoch gehen die Autoren nicht auf die Vollständigkeit der Ontologie und einer möglichen Implementierung ein. Im Gegensatz zu CASE liegt der Fokus der Arbeit zudem nicht auf der Repräsentation digitaler Beweismittel.

Ontologien fanden in vielen weiteren Forschungsarbeiten der digitalen Forensik Anwendung. Beispielsweise entwickelten Harrill und Mislan [50] eine Ontologie zur Repräsentation von Wissen über digitale Kleingeräte („small scale digital devices“) wie Mobiltelefone, persönliche digitale Assistenten und Softwarekomponenten. Allerdings beschrieben die Autoren hauptsächlich die entsprechende Domäne, aber kein konkretes Design der Ontologie. Eine weitere Arbeit stellte eine OWL-Ontologie mit der Bezeichnung Parallax Forensics (ParFor) vor, um verschiedene Datenquellen einheitlich dar-

zustellen [51]. Ziel der Autoren ist hierbei die Identifikation, Extraktion und Speicherung von Daten, die sich auf Entitäten und ihre Beziehungen innerhalb der Domäne der digitalen Forensik beziehen. Der Fokus liegt insbesondere auf Dateisystemhierarchien, Verzeichnisse, Dateien, Berechtigungen für Ordner und Dateien, Systemereignisse, Systeminformationen, installierte Software, Benutzerkonten sowie Benutzerereignisse. ParFor erlaubt darüber hinaus zur Unterstützung von Ermittlern den Einsatz eines regelbasierten Systems, um Daten aus mehreren Quellen automatisch extrahieren und Events ableiten zu können. Ein anderes Beispiel ist die Web Services Forensic Ontology (WSFO), welche zur Repräsentation von Webservice-Protokolldaten verwendet werden kann [52]. Dazu nutzt sie ebenfalls OWL und besteht aus acht individuellen Modulen: Vorfall, Hard- und Software, Qualifikation von Einzelnen, Sicherheit, Attribution, Webdienste, Beweismittelkette sowie Gewichtung. Zweck der Ontologie ist laut den Autoren, verborgene Informationen zu identifizieren und mögliche Szenarien von Cyberangriffen aus den Webprotokollen zu ermitteln. Für forensische Untersuchungen in sozialen Netzwerken wurde in [53] die Event-based forensic Integration Ontology for Online Social networks (EFIOSN) beschrieben. Das Modell stellt insbesondere die Korrelationen zwischen Events und Entitäten aus sozialen Netzwerken dar und ermöglicht eine automatisierte Extraktion, Analyse und Interpretation von großen Massen forensischer Daten aus dieser Domäne. Die Autoren planen in Zukunft EFIOSN um plattformspezifische Ontologien zu erweitern.

Im Bereich der Smartphone-Forensik beschäftigten sich bereits Alzaabi et al. in [54] mit dem Einsatz von Ontologien. Dabei stellten die Autoren Forensic-Driven Ontologies for Smartphones (F-DOS), eine Menge von Ontologien zur Modellierung von Smartphone-Inhalten im Rahmen forensischer Untersuchungen, vor. F-DOS fungiert als abstrakte Schicht zwischen den extrahierten Smartphone-Daten und dem durch den Ermittler nutzbaren Interface. Die enthaltenen Ontologien werden in eine Kernontologie sowie mehrere Domänenontologien unterteilt. Die Kernontologie ist die Hauptontologie von F-DOS und stellt mit den Klassen *DataObject*, *EvidenceElement* sowie *DataSource* drei Hauptkonzepte bereit. Somit können Daten, Beweise und Datenquellen mitsamt ihrer gegenseitigen Beziehungen repräsentiert werden. Darüber hinaus verknüpft die Kernontologie über die Klasse *EvidenceElement* alle Domänenontologien miteinander. Eine Domänenontologie wird zur genaueren Spezifizierung einer bestimmten Domäne verwendet. Unter anderem existieren derartige Ontologien für Kontaktinformationen, Nachrichten, E-Mails, Kalendereinträge, Standortdaten, Dateien, Events und Informationen bezüglich des untersuchten Falls. F-DOS ermöglicht laut den Autoren den Aufbau einer Wissensbasis, an die Anfragen über SPARQL gestellt werden kann. Ebenfalls lassen sich mithilfe von Regeln logische Schlussfolgerungen ziehen. Allerdings gaben die Autoren nicht an, ob und wo F-DOS öffentlich zur Verfügung gestellt wird.

In der Wissenschaft existiert ebenfalls eine Reihe an Veröffentlichungen, die CASE oder vorherige Projekte direkt behandeln [55–59]. In [55] wird CybOX als Open-Source-Grundlage zum Speichern und Austausch digitaler forensischer Informationen beschrieben. Somit lassen sich Objekte und Beziehungen repräsentieren, die in digitalen Untersuchungen auftreten. Zusätzlich definieren die Autoren das Schema und die Ontologie mit der Bezeichnung Digital Forensic Analysis eXpression (DFAX), wobei DFAX als Schicht über CybOX fungiert, um weitere domänenspezifische Informationen zur Verfügung zu stellen. Dadurch wird unter anderem die Abbildung forensisch relevanter Aktionen ermöglicht. Darüber hinaus gibt die Arbeit eine erste Einführung in die UCO. Mithilfe dieser Ontologie sollen Konzepte, die in der gesamten Cyberdomäne auftreten, ausgedrückt werden können. CASE wurde in [56] als Weiterentwicklung von DFAX vorgestellt. Laut den Autoren kann CASE in jedem Kontext von Kriminalität, Unternehmen und Geheimdiensten verwendet werden und ermöglicht die Kombination von Informationen aus verschiedenen Organisationen, Datenquellen und

forensischen Tools. CASE orientiert sich an der UCO und erweitert sie, um insbesondere verschiedene Arten von Spuren und ihren Kontext repräsentieren zu können. Des Weiteren betonen die Autoren, dass das entwickelte Format hauptsächlich zum Ausdruck sowie Austausch von Informationen aus Cyberuntersuchungen dient und in erster Linie nicht als internes Datenmodell für Datenbanken oder Anwendungen gedacht ist. Zusammengefasst liefert die beschriebene Arbeit folgende Beiträge [56]:

- Offene, von einer Community entwickelte Spezifikationssprache und Ontologie mit einer API-Implementierung (Proof-of-Concept) und Beispielen für die Verwendung von CASE zur Unterstützung des Informationsaustauschs und der Interoperabilität von Tools
- Angleichung von Ontologie und Datenstrukturen an bestehende forensische Systeme und Tools zur Erleichterung der Implementierung und Übernahme durch die jeweiligen Entwickler
- Flexibles und leicht erweiterbares Datenmodell zur Darstellung beliebiger Cyberinformationen und ihrer Eigenschaften
- Formalisierte Mechanismen zur Kategorisierung und Kommentierung von Spuren und Aktionen
- Verwendung von JSON-LD als Standardserialisierung zur Unterstützung der vollständigen strukturellen und semantischen Validierung

In Ergänzung zu [56] enthält [58] mehrere Korrekturen und Aktualisierungen bezüglich der gegebenen Beispiele. In einem Buchkapitel [57] stellen die Autoren ausführlich UCO und CASE ähnlich wie in [56] vor. Darüber hinaus geben sie Leitsätze, welche aus den Erfahrungen von DFAX, CybOX und weiteren Projekten gezogen wurden, zur Strukturierung und zum Austausch von Informationen über digitale Untersuchungen an. Daraus folgend wurde bei der Entwicklung auf Expressivität, Integration statt Duplizierung, Flexibilität, Erweiterbarkeit sowie Automatisierbarkeit geachtet. In den nächsten Entwicklungsschritten möchten die Autoren UCO und CASE erweitern, sodass zum Beispiel gleiche Muster zwischen unterschiedlichen Fällen gesucht werden können. Somit könnten sich effektive Lösungsansätze wiederverwenden und Verbindungen zwischen Straftaten aufdecken lassen. Eine weitere Arbeit beschreibt, wie mittels CASE die Ergebnisse von Dateiwiederherstellungsoperationen klassifiziert, authentifiziert, bewertet und präsentiert werden können [59]. Dadurch erhöhen sich laut den Autoren die Klarheit und Konsistenz bezüglich der Ergebnisse, was Ermittlern hilft den Kontext sowie die Zuverlässigkeit der durchgeführten Operationen zu verstehen. Außerdem unterstützt die Ontologie bei der Ergebnisbewertung und verringert die Wahrscheinlichkeit von Fehlern.

Zusammengefasst existiert eine Vielzahl an Forschungsarbeiten bezüglich der Verwendung von Ontologien im Bereich der digitalen Forensik. Derartige Wissensrepräsentationen werden immer notwendiger, um die aktuellen und zukünftig aufkommenden Herausforderungen bestehen zu können. Insbesondere standardisierte Austauschformate für Tools und mehr Automatisierungen im Untersuchungsprozess müssen dafür geschaffen werden. Im Rahmen dieser Arbeit kommt eine Auswahl der vorgestellten Ontologien in Betracht, um eine Subdomäne für forensische Untersuchungen mobiler Kommunikation zu modellieren. Hierbei wurde die CASE-Ontologie gewählt, da sie aus einem aktuellen und regelmäßig gepflegten Projekt stammt und von einem großen, namhaften Konsortium von Forschungsinstituten, staatlichen Einrichtungen sowie Softwareunternehmen entwickelt wird. Deshalb besteht die Chance, dass CASE zukünftig von der überwiegenden Mehrheit der Forscher, IT-Forensiker und Entwickler forensischer Software akzeptiert oder verwendet wird und sich CASE somit in Zukunft als ein Standard etablieren kann.

4 Methoden

4.1 Abbildung des MoNA-Modells auf die CASE-Ontologie

Bei der Erstellung der Ontologie zur Repräsentation mobiler Kommunikation richtete sich diese Arbeit nach der Vorgehensweise, die der „Mapping Guide“ des CASE-Konsortiums [60] beschreibt. Dabei wurden folgende Schritte durchgeführt:

1. Bewertung von MoNA bezüglich des Zwecks und Ziels
2. Identifizierung von abzubildenden Klassen des MoNA-Datenmodells
3. Nutzen vorhandener Beispiele, wenn möglich
4. Vergleich der festgestellten Klassenterminologie mit dem CASE-Vokabular
5. Überprüfung der semantischen Repräsentation der jeweiligen MoNA-Klasse und der CASE-Klasse sowie Feststellung, ob 1-zu-1-Abbildung besteht
6. Verwendung des CASE-Arbeitsblatts zum Erhalt abgebildeter Klassen

Bei der Durchführung des dritten Schritts wurden die Beispiele von der CASE-Website[12] und aus dem CASE-Repository[13] auf GitHub ausgewertet. Hierbei wurde geprüft, ob sich diese Beispiele für die Abbildung von MoNA- auf CASE-Klassen adaptieren lassen.

Im Detail orientierte sich diese Arbeit bei den aufgeführten Schritten nach dem Flussdiagramm, welches im „Mapping Guide“ des CASE-Konsortiums [60] gegeben ist und den Abbildungsprozess ausführlich veranschaulicht. Dementsprechend wurden zuerst die abzubildenden Klassen des MoNA-Datenmodells aufgelistet. Dabei fokussierte sich diese Arbeit nur auf Klassen und deren Attribute, welche zur Modellierung mobiler Kommunikation dienten. Klassen und Attribute, die MoNA spezifische Annotationen und Ergebnisse repräsentierten, wurden nicht berücksichtigt. Somit wurde im Rahmen dieser Arbeit eine generalisierte Ontologie zur Repräsentation mobiler Kommunikation erarbeitet, die ebenso andere Tools nutzen können. Falls notwendig, wurden die Klassen nach der erfolgten Auflistung gruppiert, sodass sie ähnliche Daten repräsentieren. Daraufhin wurden für jede Klasse Konzepte, welche deren jeweilige Semantik repräsentieren, in CASE gesucht. Dazu wurden die Turtle-Dateien der CASE-Beispiele per `Strg+F` durchsucht. Eine Suche in der aktuellsten Dokumentation von der UCO[14] und CASE[15] ergänzte diesen Schritt. Falls kein passendes Konzept zur Abbildung einer MoNA-Klasse gefunden werden konnte, wurde die entsprechende Klasse als „Lücke“ markiert. Anderenfalls erfolgte die Überprüfung, ob die Klasse nur ein Teil des Konzeptes ist und mit anderen Klassen zusammengeführt werden muss. Beim Eintreten des Falls wurde die Klasse dementsprechend markiert. Ansonsten konnte eine mögliche 1-zu-1-Abbildung gefunden werden, was ebenfalls dokumentiert wurde. Dabei war es möglich, dass mehrere geeignete Konzepte gefunden werden konnten. Der bisher geschilderte Prozess ist in Abbildung 4.1 als Flussdiagramm dargestellt.

[12] CASE-Beispiele auf der CASE-Website: `https://caseontology.org/examples/`, besucht am 08.10.2023
[13] CASE-Beispiele auf Github: `https://github.com/casework/CASE-Examples`, besucht am 08.10.2023
[14] UCO-Dokumentation: `https://ontology.unifiedcyberontology.org/`, besucht am 08.10.2023
[15] CASE-Dokumentation: `https://ontology.caseontology.org/`, besucht am 08.10.2023

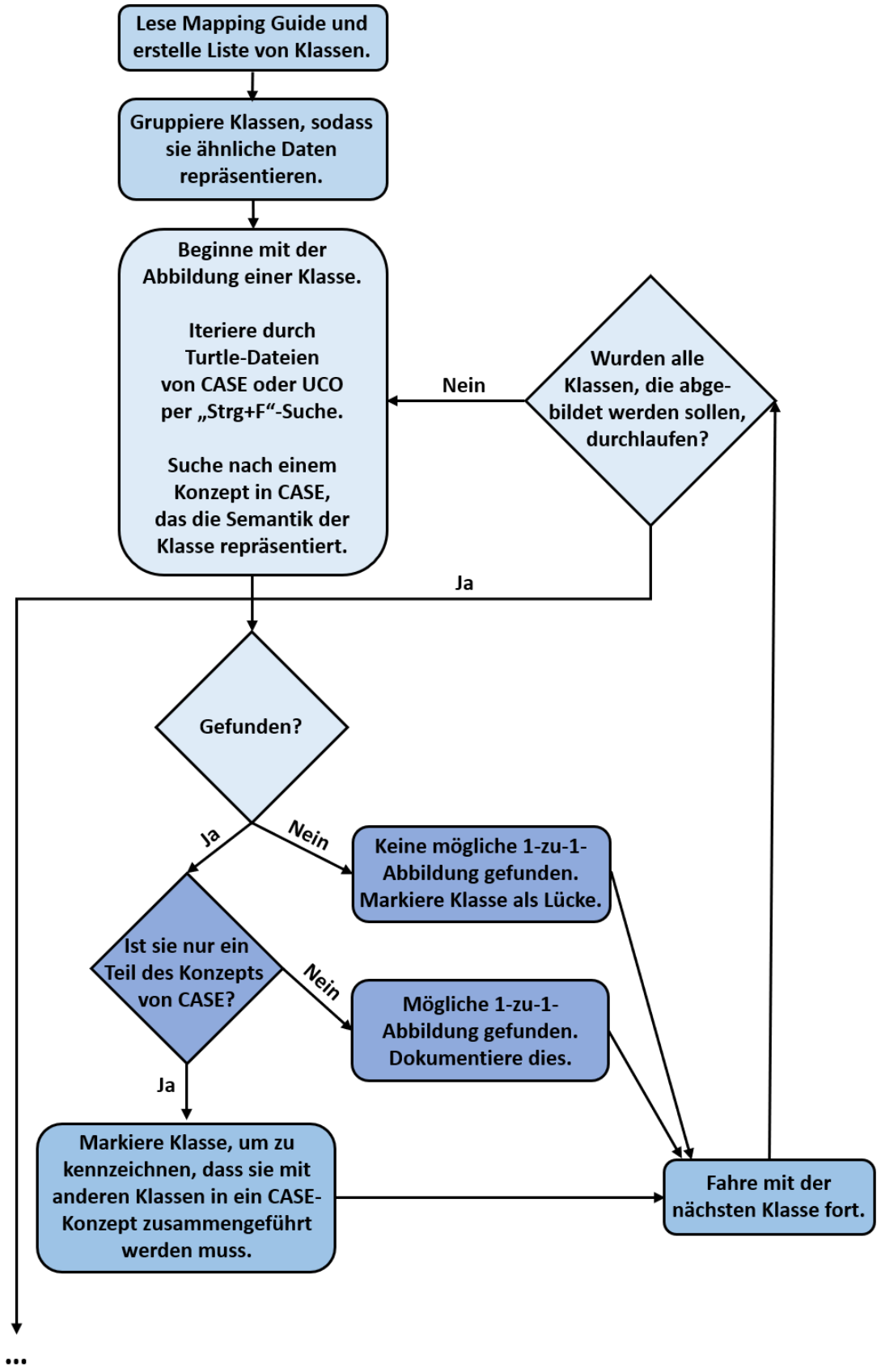

Abbildung 4.1: Erster Teil des Prozesses zur Abbildung von Klassen eines Tools auf die CASE-Ontologie nach dem „Mapping Guide“ von CASE. Adaptiert aus [60, S. 15].

Nach den beschriebenen Schritten erfolgte die Auswertung der Markierungen und Dokumentationen. Falls bei 1-zu-1-Abbildungen für eine Klasse mehrere passende Konzepte existierten, wurden die Konzepte priorisiert, wie genau sie eine Klasse abbilden. Das Konzept mit der höchsten Priorität wurde für die Abbildung gewählt. In dem Fall, dass mehrere Klassen zu einem Konzept zusammengeführt werden müssen, wurden sie jeweils auf das entsprechende Konzept abgebildet. Markierte Lücken bedurften der spezifischen Definition eines neuen `Facets`, was das CASE-Konsortium unter anderem in ihren FAQ[16] empfehlen. Bei der Vorgehensweise dieser Arbeit wurde zudem die Tabelle aus dem Arbeitsblatt des „Mapping Guides“ [60] adaptiert, um die mögliche Abbildung von MoNA-Instanzen auf CASE-Instanzen zu dokumentieren. Hierbei wurde für jede MoNA-Klasse eine Tabelle erstellt, welche die jeweils zugehörigen Attribute zwischen MoNA und CASE abbildet. Derartige Tabellen besitzen dabei je drei Spalten:

- MoNA-Attribut: Name, Datentyp und Kardinalität des Attributs in MoNA
- CASE-Klasse: Name der korrespondierenden Klasse in CASE
- CASE-Eigenschaft: Name, Datentyp und Kardinalität der Eigenschaft in CASE, welche dem Attribut aus MoNA zugeordnet ist

In der praktischen Umsetzung der Abbildung des MoNA-Datenmodells auf CASE wurde eine Java-Funktion in MoNA zum Export der Ontologie implementiert. Dabei überführt die Funktion die Daten der jeweiligen Instanzen in das JSON-LD-Format entsprechend der Anleitung für Instanzdaten [21]. JSON-LD ist ein auf JSON basierendes Format zur Serialisierung und Übertragung von verknüpften Daten, welche zum Beispiel ursprünglich im RDF-Format vorliegen können [61]. Des Weiteren wurden entsprechend der Anleitung für Instanzdaten die Namensräume im Kontext-Dictionary definiert. Der Namensraum für die CASE-Ontologie ist `https://ontology.caseontology.org/case/`, während auf die UCO mittels `https://ontology.unifiedcyberontology.org/uco/` referenziert werden kann [32]. Zur eindeutigen Identifizierung wird jeder einzelnen Instanz eine ID zugeordnet. Diese Zeichenkette setzt sich entsprechend der Anleitung für Instanzdaten [21] aus dem Namen des Instanztyps und einer UUID zusammen. Nach der Anleitung kann in CASE die UUID durch einen Zufallszahlengenerator oder durch das Hashen der Eingabedaten erzeugt werden. Als Hashverfahren werden dafür MD5 oder SHA1 empfohlen. In dieser Arbeit wurde in der prototypischen Exportfunktion in MoNA der Wert des Attributs `id` als UUID für die jeweilige Instanz verwendet. Falls der entsprechende Wert nicht vorhanden war, wurde in diesem Fall der Hashwert, den die Java-Methode `hashCode()` der jeweiligen MoNA-Instanz zurückgibt, genutzt. In Ausnahmefällen, in denen weder der Hashwert noch die `id` verwendet werden konnte, wurde aus dem Wert eines Attributs, das eine Instanz eindeutig identifiziert, mithilfe des MD5-Algorithmus [62] ein Hashwert erzeugt. Der entsprechende Wert ließ sich daraufhin als UUID nutzen. Für eine Instanz, welche eine Beziehung darstellt, diente ebenfalls ein Hashwert als UUID. Dieser Wert wurde aus den verketteten UUIDs der zugehörigen Quell- und Zielinstanz durch Anwendung des MD5-Verfahrens gebildet.

Das Datenmodell in MoNA basiert auf dem Eclipse Modeling Framework (EMF). Dieses Framework stellt eine Menge von Plug-ins bereit, die das Modellieren des Datenmodells sowie die Generierung von Code aus dem Modell ermöglichen [63]. Eine *ecore*-Datei enthält dabei die Informationen über die definierten Klassen, ihre Attribute sowie deren gegenseitige Beziehungen. Diese Datei wurde im Rahmen dieser Arbeit zum Verständnis des MoNA-Datenmodells in der Entwicklungsumgebung

[16] FAQ von CASE: `https://caseontology.org/resources/faq.html`, besucht am 08.10.2023

Eclipse gründlich ausgewertet. Zusätzlich wurde der grafische Modellierer *EcoreTools : Ecore Diagram Editor*[17] installiert. Damit konnte ein Klassendiagramm aus der *ecore*-Datei erstellt werden, was die Analyse des Datenmodells vereinfachte.

Zusätzlich wurde zum Verständnis von CASE die gesamte Ontologie von GitHub[18] heruntergeladen und anschließend in dem Programm *Protégé*[19] analysiert. Protégé ermöglicht die Erstellung und Wartung von Ontologien [64]. Zum Einlesen der gesamten CASE-Ontologie musste dafür die Turtle-Datei `case.ttl` aus dem Verzeichnis `/CASE/ontology/master/` in dem Programm geöffnet werden. Mithilfe des Plug-ins *OwlViz*[20] konnte insbesondere die Klassenhierarchie von CASE in Protégé grafisch veranschaulicht werden. Zur Verwendung von OwlViz musste zuvor *GraphViz*[21] separat installiert werden. Im Anschluss konnte OwlViz in Protégé über die Auswahl `Window -> Tabs -> OwlViz` aktiviert werden.

4.2 Aufsetzen eines semantischen Webservers

Für die prototypische Umsetzung eines semantischen Webservers wurde eine virtuelle Maschine auf einem Windows-System aufgesetzt. VirtualBox[22] in der Version 7.0.6 diente hierfür als Virtualisierungssoftware. Unter `https://www.osboxes.org/ubuntu/` wurde das Linux-Image *Ubuntu 22.10 Kinetic Kudu* heruntergeladen und in VirtualBox installiert. Daraufhin wurde *Apache Tomcat*[23] auf dem virtualisierten Ubuntu-System eingerichtet. Dazu wurden die aus [65] adaptierten Befehle, die in Quelltext 4.1 aufgeführt sind, in der Konsole ausgeführt.

```
1  sudo apt update # Aktualisierung der Ubuntu-Repositories
2
3  sudo apt install default-jdk # Installation des aktuellsten OpenJDKs
4
5  java -version # Anzeige der installierten Java-Version
6
7  # Anlegen eines Nutzers zur Verwendung des Tomcat-Services
8  sudo useradd -r -m -U -d /opt/tomcat -s /bin/false tomcat
9
10 # Herunterladen des Tomcat-Pakets
11 wget -c https://downloads.apache.org/tomcat/tomcat-8/v8.5.91/bin/apache-
     tomcat-8.5.91.tar.gz
12
13 # Entpacken des Tomcat-Pakets
14 sudo tar xf apache-tomcat-8.5.91.tar.gz -C /opt/tomcat
15
16 # Anlegen eines symbolischen Links auf Tomcat-Verzeichnis
17 sudo ln -s /opt/tomcat/apache-tomcat-8.5.91 /opt/tomcat/updated
18
19 # Gewährung des Zugriffs auf Tomcat-Verzeichnis für Tomcat-Nutzer
20 sudo chown -R tomcat: /opt/tomcat/*
21
```

[17] Eclipse Marketplace-Seite für EcoreTools - Ecore Diagram Editor: `https://marketplace.eclipse.org/content/ecoretools-ecore-diagram-editor`, besucht am 09.10.2023
[18] CASE-Ontologie auf GitHub: `https://github.com/casework/CASE`, besucht am 09.10.2023
[19] Website von Protégé: `https://protege.stanford.edu/`, besucht am 09.10.2023
[20] OwlViz auf GitHub: `https://github.com/protegeproject/owlviz`, besucht am 09.10.2023
[21] Website von GraphViz: `https://www.graphviz.org/`, besucht am 09.10.2023
[22] Website von VirtualBox: `https://www.virtualbox.org/`, besucht am 09.10.2023
[23] Website von Apache Tomcat: `https://tomcat.apache.org/`, besucht am 09.10.2023

```
22  # Vergabe aller Ausführungsrechte für alle Tomcat-Skripte in bin/
23  sudo sh -c 'chmod +x /opt/tomcat/updated/bin/*.sh'
24
25  # Erstellung der Tomcat-Service-Datei
26  sudo nano /etc/systemd/system/tomcat.service
27
28  sudo systemctl daemon-reload # Neuladen des Daemons
29
30  sudo systemctl start tomcat # Starten des Tomcat-Services
31
32  sudo systemctl enable tomcat # Auto-Start des Services bei Systemstart
33
34  sudo ufw allow 8080/tcp # Erlaubnis der Verwendung von Port 8080
```

Quelltext 4.1: Konsolenbefehle zur Einrichtung eines Apache Tomcat-Servers auf einem Ubuntu-System.

Zuerst musste eine aktuelle Version des OpenJDK, das Java-Entwicklungspaket, installiert werden. Hierfür wurden die Ubuntu-Repositories aktualisiert (Quelltext 4.1, Zeile 1) und danach die neueste Version des OpenJDK auf dem Ubuntu-System installiert (Quelltext 4.1, Zeile 3). Mithilfe des Befehls in Zeile 5 im Quelltext 4.1 konnte die installierte Version überprüft werden. In diesem Fall befand sich das OpenJDK in der Version 11.0.19. Im Anschluss wurde ein Tomcat-Nutzer auf dem System angelegt (Quelltext 4.1, Zeile 8). Der Tomcat-Service sollte demzufolge im Verzeichnis `/opt/Tomcat` verwaltet werden. Die Erstellung des Nutzers erfolgte unter anderem aus Sicherheitsgründen, da der Tomcat-Service somit unabhängig vom Root-Account des Systems laufen konnte. Als Nächstes wurde das Tomcat-Paket in der Version 8.5.91 heruntergeladen (Quelltext 4.1, Zeile 11). Danach wurde das Paket in dem gewünschten Verzeichnis entpackt (Quelltext 4.1, Zeile 14). Für das Tomcat-Verzeichnis wurde ein symbolischer Link angelegt (Quelltext 4.1, Zeile 17). Tomcat kann auf dem System einfacher aktualisiert werden, da der symbolische Link in den nächsten Schritten der Konfiguration als Referenz genutzt wurde. Somit muss für neue Tomcat-Versionen nur der symbolische Link anstelle der gesamten Konfiguration angepasst werden. Durch den Befehl in Zeile 20 im Quelltext 4.1 wurde dem Tomcat-Nutzer Zugriff auf das Installationsverzeichnis von Tomcat gewährt. Anschließend wurden allen Skripten im Verzeichnis `bin` des Tomcat-Verzeichnisses alle Ausführungsrechte erteilt (Quelltext 4.1, Zeile 23). Daraufhin folgte die Erstellung der Konfigurationsdatei für den Tomcat-Service (Quelltext 4.1, Zeile 26). Hierbei wurden unter anderem die Umgebung definiert, Skripts für den Start und das Beenden des Servers angegeben sowie administrative Nutzer und Gruppen festgelegt. Der vollständige Inhalt der Konfigurationsdatei ist in Quelltext 4.2 aufgeführt. Für die weitere Vorgehensweise war insbesondere wichtig, die Umgebungsvariable `FUSEKI_BASE` zu definieren (Quelltext 4.2, Zeile 14). Dabei wurde der Verzeichnispfad angegeben, unter dem zukünftig alle Ontologien und damit verbundene Daten gespeichert werden sollen. Dabei war zu beachten, dass der Tomcat-Nutzer Schreibrechte für dieses Verzeichnis besitzen muss. Ohne die explizite Definition der Variable `FUSEKI_BASE` wäre sie standardmäßig `/etc/fuseki/`. Unter Linux besitzt jedoch nur der Root-Account Schreibrechte für das Verzeichnis `/etc/`. Deshalb hätte der Server mit den Rechten des Tomcat-Nutzers keine Berechtigungen, um Datensätze erstellen und speichern zu können.

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment="JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64"
Environment="CATALINA_PID=/opt/tomcat/updated/temp/tomcat.pid"
Environment="CATALINA_HOME=/opt/tomcat/updated/"
Environment="CATALINA_BASE=/opt/tomcat/updated/"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"
Environment="JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/
  dev/./urandom"
Environment="FUSEKI_BASE=/home/osboxes/fuseki/base"

ExecStart=/opt/tomcat/updated/bin/startup.sh
ExecStop=/opt/tomcat/updated/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

Quelltext 4.2: Inhalt der Konfigurationsdatei eines Tomcat-Services auf einem Ubuntu-System.

Nach der Konfiguration des Tomcat-Services wurde der Daemon neu geladen, um das System zu aktualisieren (Quelltext 4.1, Zeile 28). Danach konnte der Tomcat-Server gestartet werden (Quelltext 4.1, Zeile 30). Damit mit jedem Systemstart auch der Server gestartet wird, wurde der Befehl in Zeile 32 im Quelltext 4.1 ausgeführt. Mithilfe des Befehls `sudo systemctl status tomcat` konnte der Serverstatus abgerufen und die Funktionstüchtigkeit geprüft werden. Weiterhin wurde die Verwendung von Port 8080 erlaubt, um den Tomcat-Server über diesen Port zur Verfügung zu stellen (Quelltext 4.1, Zeile 34). Abschließend wurde die Konfigurationsdatei für Tomcat-Nutzer modifiziert, um den Nutzernamen und das Passwort für den Administrator festzulegen. Mithilfe des Konsolenbefehls `sudo nano /opt/tomcat/updated/conf/tomcat-users.xml` wurde die entsprechende Datei geöffnet und der Inhalt durch die in Quelltext 4.3 aufgeführten Zeilen ersetzt. Für die prototypische Umsetzung wurde der Nutzername `admin` und das Passwort `admin` gewählt (Quelltext 4.3, Zeile 9). Im Praxiseinsatz müsste hierbei ein sicheres Passwort festgelegt werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users>
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<user username="admin" password="admin" roles="manager-gui, manager-script,
   manager-jmx, manager-status, admin-gui, admin-script"/>
</tomcat-users>
```

Quelltext 4.3: Inhalt der Konfigurationsdatei für Tomcat-Nutzer auf einem Ubuntu-System.

Mithilfe des Web-Application-Managers von Tomcat wurde *Apache Jena Fuseki* aufgesetzt. Dabei handelt es sich um einen SPARQL-Server, der entweder als Betriebssystem-Service, Java-Webapplikation oder als Standalone-Server betrieben werden kann [66]. Im Rahmen dieser Arbeit wurde die Java-Webapplikation verwendet. Dazu wurde die entsprechende WAR-Datei für Apache Jena Fuseki von der Downloadseite[24] heruntergeladen. Für diese Arbeit fand Apache Jena Fuseki in der Version 4.7.0 Anwendung. Die WAR-Datei wurde zu `fuseki.war` umbenannt und anschließend in der grafischen Benutzeroberfläche des Web-Application-Managers von Tomcat ausgewählt und angewendet. Den Verzeichnispfad, der als Umgebungsvariable `FUSEKI_BASE` definiert wurde, verwendet der Fuseki-Server zum Anlegen und Speichern von Daten. Dieser Server war nach Durchführung der aufgeführten Schritte innerhalb des Ubuntu-Systems unter `http://localhost:8080/fuseki/#/` verfügbar. Um auf den Server von Seiten des Windows-Systems, auf dem VirtualBox installiert war, zugreifen zu können, musste in den VirtualBox-Einstellungen für das virtualisierte Ubuntu-System unter dem Menüpunkt „Netzwerk“ `Angeschlossen an: Host-only Adapter` ausgewählt werden (siehe Abbildung 4.2).

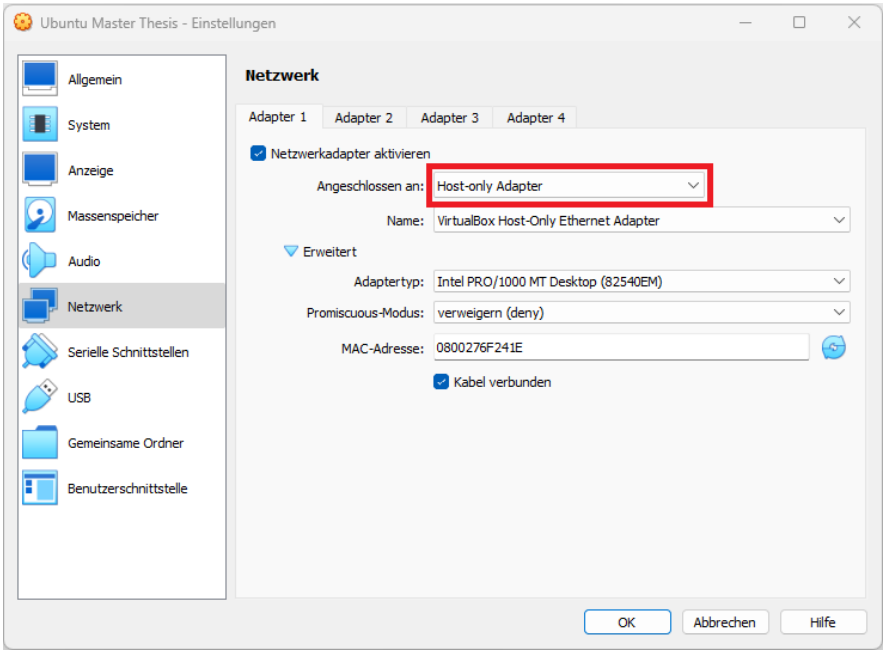

Abbildung 4.2: Netzwerkeinstellungen für das Ubuntu-System in VirtualBox. Wichtig war hierbei die Auswahl `Angeschlossen an: Host-only Adapter`, damit das Host-System auf den vom Ubuntu-System zur Verfügung gestellten semantischen Webserver zugreifen konnte.

Durch Ausführung des Konsolenbefehls `hostname -I` unter Ubuntu konnte die IP-Adresse ermittelt werden, unter welcher der Fuseki-Server für das Windows-System verfügbar ist. Diese IP-Adresse musste in `http://<IP-Adresse>:8080/fuseki/#/` anstelle von `<IP-Adresse>` eingesetzt werden, um die vollständige URL zum Aufrufen des Servers zu erhalten. Für die Sicherheit der Fuseki-Webapp wurde die automatisch erzeugte Standard-Konfiguration von *Apache Shiro*[25] verwendet. Allerdings musste die Konfiguration per `nano ~/fuseki/base/shiro.ini` angepasst werden, um den Zugriff sowie das Erstellen und Bearbeiten von Datensätzen für das Windows-System zu erlauben. Standardmäßig dürfte durch die Zeile `/$/** = localhostFilter` in der Initialisierungsdatei nur der Server-Host, in diesem Fall das virtualisierte Ubuntu-System, diese Operationen durchführen.

[24] Apache Jena Releases: `https://jena.apache.org/download/index.cgi`, besucht am 09.10.2023
[25] Sicherheit in Fuseki: `https://jena.apache.org/documentation/fuseki2/fuseki-security.html`, besucht am 10.10.2023

Deshalb wurde die entsprechende Zeile auskommentiert und die Zeile `/$/** = anon` hinzugefügt, wodurch diese Operationen für alle Nutzer erlaubt wurden. Hierbei ist bei dieser prototypischen Implementierung anzumerken, dass der Fuseki-Server durch die Einstellungen der Virtualisierung nur für das Windows- und das virtualisierte Ubuntu-System verfügbar war.

Beim Erstellen eines Datensatzes in der Weboberfläche des Fuseki-Servers konnte ausgewählt werden, ob der Inhalt des Datensatzes persistiert oder nur bis zum Serverneustart im Speicher abgelegt werden soll. Für das Persistieren von Daten wird das Speichersystem *TDB2*[26] verwendet, welches die Daten unter dem Verzeichnis `~/fuseki/base/databases/` in der prototypischen Implementierung des Ubuntu-Systems ablegt und organisiert.

4.3 Anwendungstest der entworfenen Ontologie

Zum Testen der entworfenen Ontologie und der in MoNA implementierten Exportfunktion wurde eine Reihe von Kommunikationsdaten zur Verfügung gestellt. Diese Testdaten wurden auf zwei Smartphones generiert und mithilfe der UFED Touch2-Plattform physisch gesichert. Bei den beiden Geräten handelte es sich um ein Samsung Galaxy S9 SM-G960F mit der Android-Version 10 sowie ein Apple iPhone 7 (A1778) mit der iOS-Version 14.7.1. Auf jedem Smartphone waren die Messenger-Dienste WhatsApp, Telegram und Facebook Messenger installiert. Unter jedem Messenger wurden hierbei mehrere Nachrichten in 1-zu-1-Chats und Gruppenchats ausgetauscht. Die Nachrichteninhalte bestanden aus Text und Multimediadateien wie Bilder, Videos, Audios oder Dokumente. Die Kommunikationsdaten der Messenger-Dienste wurden in MoNA eingelesen. Dort konnte unter anderem eine Statistik über die enthaltenen Kommunikationen eingesehen werden, welche in Tabelle 4.1 aufgeführt ist. Im Anschluss wurde die CASE-Exportfunktion in MoNA für die eingelesenen Daten getestet.

Tabelle 4.1: Statistik der generierten Kommunikationsdaten für den Anwendungstest der entworfenen Ontologie.

	Samsung Galaxy S9	**Apple iPhone 7**	**Akkumuliert**
Kommunikationsdienste	3	3	6
Kontakte	40	8	48
Gruppen	11	9	20
Nachrichten	2968	2711	5679
Multimediadateien	81	75	156
davon:			
Bilder	39	42	81
Videos	19	11	30
Audios	22	21	43
Dokumente	1	1	2

Im Anschluss wurde die exportierte Ontologie, welche sich im JSON-LD-Format befand, ausgewertet. Hierzu wurden für jede Klasse ausschnittweise erzeugte Instanzen bezüglich ihrer Form und semantischen Korrektheit kontrolliert. Darüber hinaus erfolgte die Konvertierung des Exports von JSON-LD zu Turtle, sodass sich die Ontologie in den aufgesetzten semantischen Webserver einlesen ließ. Zum Parsen der JSON-LD-Datei wurde ein Python-Skript geschrieben, welches die

[26] Dokumentation von TDB2: https://jena.apache.org/documentation/tdb2/, besucht am 10.10.2023

Python-Bibliothek *RDFLib*[27] verwendet. Diese Bibliothek ermöglicht das Arbeiten mit RDF und stellt unter anderem Methoden zum Parsen und Serialisieren für RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, JSON-LD, HexTuples, RDFa and Microdata bereit. Zum Starten des Skripts muss Python in der Version 3.8.1 oder höher installiert sein. Zudem ist die Installation von RDFLib notwendig, was beispielsweise mithilfe des Paketverwaltungsprogramms *pip* durch Ausführung der Kommandozeile `pip install rdflib` erfolgen kann. Der Code des Skripts ist in Quelltext 4.4 gegeben. Das Skript lässt sich auf der Kommandozeile ausführen. Hierzu muss der Dateipfad der JSON-LD-Datei als erster Parameter und der gewünschte Ausgabepfad der Turtle-Datei als zweiter Parameter angegeben werden.

```
1  from argparse import ArgumentParser
2  from rdflib import Graph
3
4  if __name__ == '__main__':
5    # Definition der Kommandozeilenparameter
6    parser = ArgumentParser()
7    parser.add_argument("input", type=str, help="path of the JSON-LD file")
8    parser.add_argument("output", type=str, help="output path")
9    args = parser.parse_args()
10
11   # Parsen der JSON-LD-Datei und Serialisierung als Turtle-Datei
12   g = Graph()
13   g.parse(location=args.input, format='json-ld')
14   g.serialize(destination=args.output, format='turtle')
15
```

Quelltext 4.4: Python-Skript zum Konvertieren von JSON-LD-Dateien in das Turtle-Dateiformat.

Auf dem semantischen Webserver wurde in Fuseki ein neuer Datensatz angelegt und die Daten aus der Turtle-Datei geladen. Daraufhin wurden verschiedene Anfragen in SPARQL formuliert und an den Server gesendet. Dabei wurde insbesondere die Vollständigkeit der exportierten Daten überprüft, indem für jede Klasse angefragt wurde, wie viele entsprechende Instanzen in der Ontologie existierten. Die entsprechenden Ausgaben ließen sich daraufhin mit der Statistik aus Tabelle 4.1 vergleichen.

[27] Dokumentation von RDFLib: https://rdflib.readthedocs.io/en/stable/, besucht am 10.10.2023

5 Ergebnisse

Im Rahmen dieser Arbeit sollte eine Ontologie zur Repräsentation mobiler Kommunikation für forensische Untersuchungen entworfen werden. Dabei bot sich die Nutzung einer Subdomäne der CASE-Ontologie an. Zur Orientierung bei der Entwicklung diente das Datenmodell von MoNA, da es bereits mobile Kommunikation abbildet und die Daten mit Semantik anreichert. Dabei bot sich zugleich die Abbildung des Datenmodells auf CASE an.

5.1 Abbildung des MoNA-Datenmodells auf die CASE-Ontologie

Ziel von MoNA ist die Bereitstellung einer forensischen Analyseplattform zur effizienten Auswertung mobiler Kommunikation für Sachbearbeiter und Ermittler. Demzufolge muss die zu entwerfende Ontologie forensisch relevante Klassen, Eigenschaften und Beziehungen im Bereich der mobilen Kommunikation abbilden. Bei der Analyse des MoNA-Datenmodells wurden folgende Klassen für die Abbildung auf CASE ermittelt:

Examination. Diese Klasse repräsentiert forensische Untersuchungen und deren damit verbundene Informationen. Eine Untersuchung enthält dabei eine Menge von mobilen Endgeräten.

MobileDevice. Diese Klasse repräsentiert mobile Endgeräte und deren damit verbundene Informationen. Ein mobiles Endgerät enthält dabei eine Menge von mobilen Kommunikationsdiensten, Personendaten, Kontakten und lokalen Multimediadateien.

Person. Diese Klasse repräsentiert Personen und deren damit verbundene Informationen.

ContactId. Diese Klasse repräsentiert Kontakte beziehungsweise Accounts und deren damit verbundene Informationen.

MobileCommuniation. Diese Klasse repräsentiert mobile Kommunikationsdienste, wie Messenger-Dienste oder E-Mail-Anwendungen, und deren damit verbundene Informationen. Ein mobiler Kommunikationsdienst enthält dabei eine Menge von Chats.

Chat. Diese Klasse repräsentiert Chats und deren damit verbundene Informationen. Ein Chat enthält dabei eine Menge von Nachrichten.

Message. Diese Klasse repräsentiert Nachrichten und deren damit verbundene Informationen. Eine Nachricht kann dabei genau einen Nachrichtenanhang enthalten.

Media. Diese Klasse repräsentiert lokale Multimediadateien sowie multimediale Nachrichtenanhänge und deren damit verbundene Informationen.

Die Klassen mit ihren Attributen sind in Abbildung 5.1 als Klassendiagramm dargestellt.

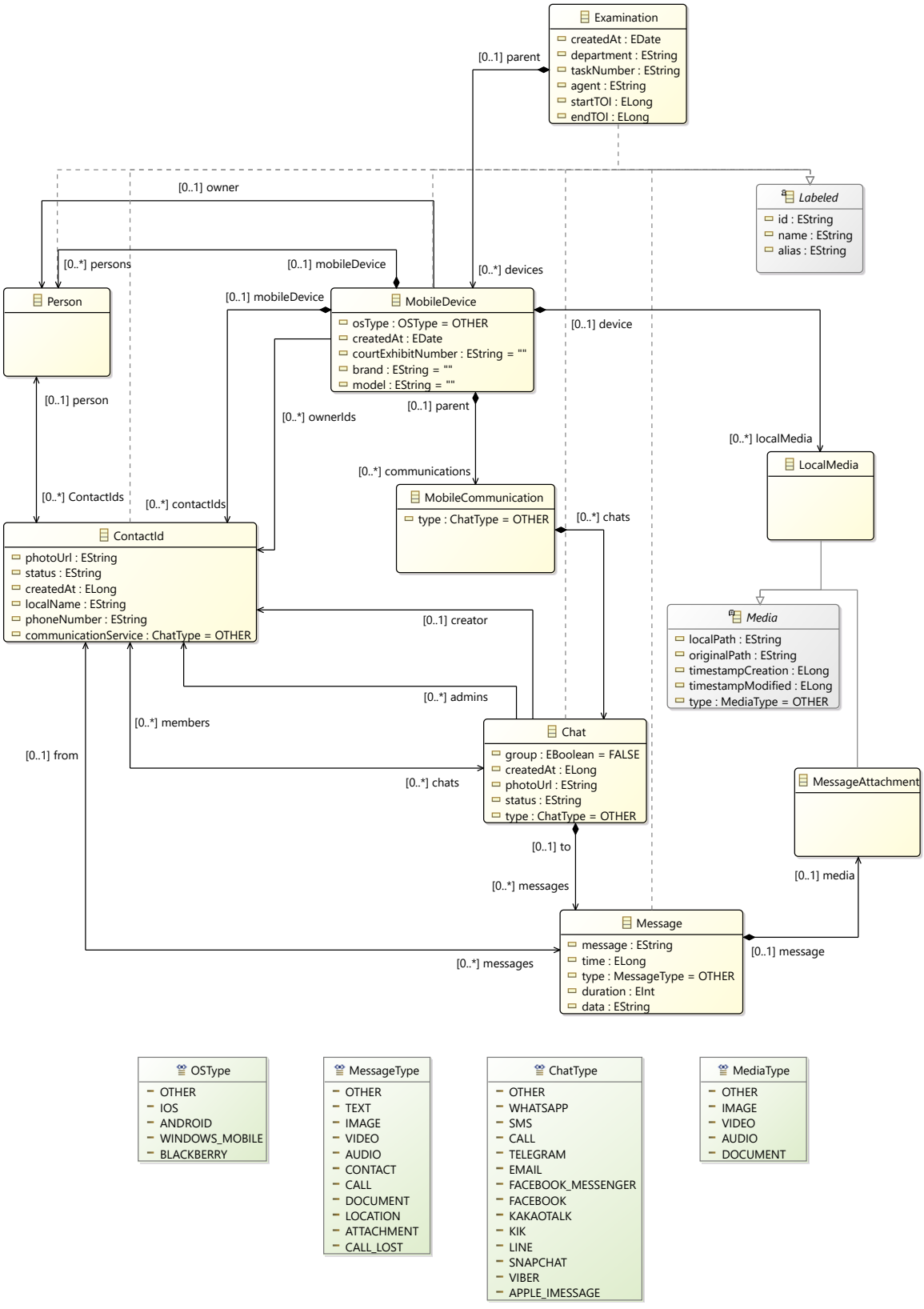

Abbildung 5.1: Klassendiagramm, welches ausgewählte Klassen und Attribute des MoNA-Datennmodells zeigt. Diese Auswahl wurde auf die CASE-Ontologie abgebildet.

In diesem gesamten Abschnitt werden die für die Ontologie verwendete IRIs folgendermaßen abgekürzt:

- investigation: https://ontology.caseontology.org/case/investigation/
- uco-core: https://ontology.unifiedcyberontology.org/uco/core/
- uco-observable: https://ontology.unifiedcyberontology.org/uco/observable/
- uco-identity: https://ontology.unifiedcyberontology.org/uco/identity/
- uco-types: https://ontology.unifiedcyberontology.org/uco/types/
- co: http://purl.org/co/
- owl: http://www.w3.org/2002/07/owl#
- xsd: http://www.w3.org/2001/XMLSchema#

Laut [33] sollte jeweils ein `uco-core:Bundle` die hierarchisch höchste Ebene für digitale Untersuchungen in CASE bilden. Alle weiteren Informationen werden in diesem `Bundle` eingebettet. Mithilfe der Klasse `uco-observable:ObservableRelationship` kann zudem die Beziehung zwischen zwei Instanzen beschrieben werden. Zur Angabe von Quell- und Zielinstanz besitzt sie dabei die Eigenschaften `uco-core:source` und `uco-core:target`. Zur Charakterisierung der Beziehung dient die Eigenschaft `uco-core:kindOfRelationship`, während über `uco-core:isDirectional` angegeben werden kann, ob die Beziehung auf die Richtung von der Quelle zum Ziel beschränkt ist. Die folgenden Unterabschnitte stellen die Abbildung der bereits aufgeführten MoNA-Klassen auf CASE dar. Falls der Wert eines MoNA-Attributs `NULL` entspricht, wird bei der Abbildung auf CASE die Eigenschaft der entsprechenden Instanz nicht angegeben.

5.1.1 Semantische Repräsentation von forensischen Untersuchungen in CASE

Eine forensische Untersuchung wird in MoNA durch die Klasse `Examination` dargestellt. In CASE kann sie mittels der Klasse `investigation:Investigation` abgebildet werden. Hierbei lässt sich die `id` einer `Examination` aus MoNA direkt als UUID der entsprechenden `Investigation` verwenden. Dabei kann eine forensische Untersuchung in MoNA zum Beispiel einen Namen (`name`), ein Aktenzeichen beziehungsweise eine Auftragsnummer (`taskNumber`) und das Erstellungsdatum (`createdAt`) als Attribut besitzen. Des Weiteren verfügt sie über ein Start- sowie Enddatum (`startTOI` und `endTOI`) zur Eingrenzung der Zeitspanne des Nachrichtenaustauschs, welche für die Ermittlungen von Interesse ist. Diese Attribute können direkt auf Eigenschaften der Klasse `Investigation` abgebildet werden, wie Tabelle 5.1 zeigt. Der Analyst oder Ermittler (in MoNA: `agent`) kann bezüglich einer Untersuchung durch die CASE-Klasse `investigation:Examiner` und insbesondere durch dessen Eigenschaft `uco-core:name` repräsentiert werden. Die zugehörige UUID entspricht dem MD5-Hashwert des Ermittler-Namens. Ein `Examiner` kann mit einer `Investigation` über eine Instanz der Klasse `ObservableRelationship` verbunden werden. Die ermittelnde Dienststelle oder Abteilung (in MoNA: `department`) lässt sich anhand einer Instanz der Klasse `uco-identity:Organization` in CASE abbilden. Der Name beziehungsweise die entsprechende Bezeichnung wird über die Eigenschaft `uco-core:name` angegeben. Deren MD5-Hashwert bildet die UUID der `Organization`-Instanz. Eine `Organization` und eine `Investigation` können durch eine Instanz der Klasse `ObservableRelationship` miteinander in Beziehung gesetzt werden. Die IDs der mobilen Endgeräte, die im Rahmen der forensischen Untersuchung ausgewertet werden, können als Werte der Eigenschaft `uco-core:object` der `Investigation` angegeben werden.

Tabelle 5.1: Abbildung der Attribute einer forensischen Untersuchung von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
name, String, 1	investigation:Investigation	uco-core:name, xsd:string, 1
createdAt, Date, 1	investigation:Investigation	uco-core:objectCreatedTime, xsd:dateTime, 1
startTOI, long, 1	investigation:Investigation	uco-core:startTime, xsd:dateTime, 1
endTOI, long, 1	investigation:Investigation	uco-core:endTime, xsd:dateTime, 1
department, String, 1	uco-identity:Organization	uco-core:name, xsd:string, 1
agent, String, 1	investigation:Examiner	uco-core:name, xsd:string, 1
taskNumber, String, 1	owl:Thing (abgeleitet)	investigation:rootExhibitNumber, xsd:string, 1

5.1.2 Semantische Repräsentation von mobilen Endgeräten in CASE

Mobile Endgeräte, die im Rahmen einer forensischen Untersuchung ausgewertet werden, lassen sich in CASE durch die Klasse `uco-observable:MobileDevice` repräsentieren. In MoNA dient hierzu die Klasse `MobileDevice`, wobei die jeweilige `id` einer derartigen Instanz direkt als UUID für eine entsprechende CASE-Instanz genutzt werden kann. Die Abbildung der MoNA-Attribute auf die CASE-Eigenschaften bezüglich mobiler Endgeräte veranschaulicht Tabelle 5.2. Der Name (`name`), das Erstellungsdatum (`createdAt`) sowie die Asservatennummer (`courtExhibitNumber`) eines `MobileDevice` in MoNA kann dabei direkt in die passenden Eigenschaften einer Instanz der Klasse `uco-observable:MobileDevice` in CASE überführt werden.

Das Hinzufügen eines `uco-observable:DeviceFacet` ermöglicht darüber hinaus die Übernahme des Gerätemodells (`model`) und der Gerätemarke (`brand`) aus MoNA. Als Typ der Marke muss dafür zusätzlich eine `uco-identity:Organization` mit der Eigenschaft `uco-core:name` definiert werden. Mithilfe des `uco-observable:OperatingSystemFacet` und dessen Eigenschaft `uco-core:name` wird zudem das Betriebssystems von mobilen Endgeräten in CASE angegeben.

Ein mobiles Endgerät kann mehrere Personendaten, Kontakte, mobile Kommunikationsdienste und lokale Medien enthalten. Die Repräsentation der entsprechenden Entitäten behandeln die Unterabschnitte 5.1.3, 5.1.4, 5.1.5 und 5.1.8. Die Beziehung zwischen einem Gerät und den aufgeführten Entitäten wird über Instanzen der Klasse `uco-observable:ObservableRelationship` abgebildet und näher durch deren Eigenschaft `uco-observable:kindOfRelationship` beschrieben.

Tabelle 5.2: Abbildung der Attribute eines mobilen Endgeräts von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
osType, OSType, 1	uco-observable: OperatingSystemFacet	uco-core:name, xsd:string, 1
createdAt, Date, 1	uco-observable:MobileDevice	uco-core:objectCreatedTime, xsd:dateTime, 1
brand, String, 1	uco-observable:DeviceFacet	uco-observable:manufacturer, uco-identity:Organization, 1
model, String, 1	uco-observable:DeviceFacet	uco-observable:model, xsd:string, 1
name, String, 1	uco-observable:MobileDevice	uco-core:name, xsd:string, 1
courtExhibitNumber, String, 1	owl:Thing (abgeleitet)	investigation:exhibitNumber, xsd:string, 1

5.1.3 Semantische Repräsentation von Personen in CASE

Eine Person wird in MoNA als eine Instanz der Klasse `Person` dargestellt. Die zugehörigen MoNA-Attribute, welche die Person bezüglich des Namens (`name`) und des Alias (`alias`) beschreiben, lassen sich direkt auf Eigenschaften der CASE-Klasse `uco-identity:Person` abbilden, wie in Tabelle 5.3 aufgeführt. Des Weiteren kann die `id` der jeweiligen MoNA-Instanz als UUID in CASE genutzt werden. Falls sie nicht vorhanden ist, wird stattdessen der Hashwert verwendet, welchen die `hashCode()`-Methode der Person-Instanz zurückgibt.

Tabelle 5.3: Abbildung der Attribute einer Person von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
name, String, 1	identity:Person	uco-core:name, xsd:string, 1
alias, String, 1	owl:Thing (abgeleitet)	uco-observable:displayName, xsd:string, 1

Für mobile Endgeräte können zudem in Form von Personen die jeweiligen Geräteeigentümer definiert werden. Hierzu bedarf es in CASE der Verknüpfung einer `Person` mit einem `MobileDevice` über eine `uco-observable:ObservableRelationship`. Das Gerät wird dabei als `uco-core:source` angeben und die Person als `uco-core:target`. Eine genauere Beschreibung der Beziehung ermöglicht die Eigenschaft `uco-observable:kindOfRelationship`, wobei in diesem Fall beispielsweise „`Has_Owner`" als deren Wert gewählt werden kann.

Außerdem lassen sich bekannte Accounts beziehungsweise Benutzerkonten zu jeder Person zuordnen, indem weitere Beziehungen mittels der Klasse `uco-observable:ObservableRelationship` definiert werden. So ist unter anderem die Aufführung aller Benutzerkonten eines Geräteeigentümers möglich. Wie Accounts in CASE repräsentiert werden können, behandelt Unterabschnitt 5.1.4.

Optional können durch die Verwendung von zusätzlichen `Facets` weitere Informationen zu einer Person hinzugefügt werden. Beispielsweise stellt das `uco-identity:SimpleNameFacet` Eigenschaften für die Definition des Vor- und Nachnamens der Person bereit. Weiterhin ist die Angabe des Geburtsdatums mithilfe des `uco-identity:BirthInformationFacet` möglich.

5.1.4 Semantische Repräsentation von Accounts in CASE

Die MoNA-Klasse `ContactId` dient zur Darstellung von Accounts beziehungsweise Benutzerkonten einer Person. In CASE kann hierzu die Klasse `uco-observable:ApplicationAccount` verwendet werden. Tabelle 5.4 veranschaulicht die Abbildung der Attribute einer `ContactId` auf die Eigenschaften eines `uco-observable:ApplicationAccount`. Als UUID in CASE kann für entsprechende Instanzen der Hashwert genutzt werden, welchen die Methode `hashCode()` der jeweiligen `ContactId` zurückgibt. Die eigentliche `id` einer `ContactId` wird für einen Account in CASE auf die Eigenschaft `uco-observable:accountIdentifier` abgebildet, welche das `uco-observable:AccountFacet` bereitstellt. Der Name (`name`), das Erstellungsdatum (`createdAt`), der Status beziehungsweise die Bio (`status`), der zugehörige Kommunikationsdienst (`communicationService`) und die Telefonnummer (`phoneNumber`) einer Instanz der Klasse `ContactId` in MoNA kann auf die passenden Eigenschaften eines `ApplicationAccount` in CASE abgebildet werden. Der Typ der CASE-Eigenschaft `uco-observable:application`, welche den vom Account verwendeten Kommunikationsdienst beschreibt, entspricht in diesem Fall der Klasse `uco-observable:Application`. Diese Klasse repräsentiert mobile Anwendungen. Über die zugehörige Eigenschaft `uco-core:name` kann die Bezeichnung der Anwendung angegeben werden.

Das `uco-observable:ContactFacet` erweitert die Eigenschaften des `ApplicationAccount`. Dadurch kann der Alias (`alias`) und lokale Name (`localName`) des Accounts aus MoNA in CASE übertragen werden. Der Alias ist der Name, den sich ein Benutzer in bestimmten Anwendungen zusätzlich neben seinem global eindeutigen Namen geben kann. Der lokale Name hingegen dient zur Abbildung des Namens, welchen ein Geräteeigentümer für den Kontakt auf seinem mobilen Endgerät vergeben hat.

Das Profilbild (in MoNA: `photoUrl`) eines Accounts kann als `uco-observable:RasterPicture` in CASE definiert werden. Hierbei bildet der MD5-Hashwert der Bilddatei die UUID. Der Dateipfad auf dem Asservat wird durch die zugehörige Eigenschaft `uco-observable:filePath` angegeben. Das Bild muss zudem über eine Instanz der Klasse `uco-observable:ObservableRelationship` mit dem jeweiligen Account verbunden werden. Als `uco-core:source` wird der `ApplicationAccount` angeben und das `RasterPicture` als `uco-core:target`. Zudem charakterisiert die Eigenschaft `uco-observable:kindOfRelationship` die Beziehung genauer, wobei in diesem Fall beispielsweise „`Has_Profile_Picture`" als Wert der Eigenschaft gewählt werden kann.

Bei Bedarf kann darüber hinaus definiert werden, ob ein Geräteeigentümer einen Account eines Kontaktes blockiert hat. Hierzu lässt sich eine `uco-observable:ObservableRelationship` verwenden. Zur konkreteren Beschreibung der Beziehung kann unter diesen Umständen beispielsweise der Wert „`Has_Blocked`" für die Eigenschaften `uco-observable:kindOfRelationship` vergeben werden. Die `Person`-Instanz des jeweiligen Geräteeigentümers entspricht dabei der Quelle (`uco-core:source`) der `ObservableRelationship`. Das entsprechende Ziel (`uco-core:target`) der Beziehung ist wiederum der blockierte `ApplicationAccount`.

Einem Account können mehrere gesendete Nachrichten und Chats, in denen er aktiv war, zugeordnet werden. Diese Verknüpfung erfolgt jeweils über Instanzen der Klasse `ObservableRelationship` in CASE. Wie die Klassen für Chats und Nachrichten modelliert werden können, wird in den Unterabschnitten 5.1.6 und 5.1.7 beschrieben.

Tabelle 5.4: Abbildung der Attribute eines Accounts von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
`id`, `String`, 1	`uco-observable:AccountFacet`	`uco-observable:accountIdentifier`, `xsd:string`, 1
`name`, `String`, 1	`uco-observable:ApplicationAccount`	`uco-core:name`, `xsd:string`, 1
`alias`, `String`, 1	`uco-observable:ContactFacet`	`uco-observable:nickname`, `xsd:string`, 0...n
`localName`, `String`, 1	`uco-observable:ContactFacet`	`uco-observable:displayName`, `xsd:string`, 1
`phoneNumber`, `String`, 1	`owl:Thing` (abgeleitet)	`uco-observable:phoneNumber`, `xsd:string`, 1
`createdAt`, `Date`, 1	`uco-observable:ApplicationAccount`	`uco-core:objectCreatedTime`, `xsd:dateTime`, 1
`photoUrl`, `String`, 1	`uco-observable:RasterPicture`	`uco-observable:filePath` `xsd:string`, 1
`status`, `String`, 1	`owl:Thing` (abgeleitet)	`uco-core:description`, `xsd:string`, 1
`communicationService`, `ChatType`, 1	`owl:Thing` (abgeleitet)	`uco-observable:application`, `uco-observable:Application`, 1

5.1.5 Semantische Repräsentation von mobilen Kommunikationsdiensten in CASE

Mobile Kommunikationsdienste werden in MoNA durch die Klasse `MobileCommunication` beschrieben. Diese Klasse lässt sich in CASE als `uco-observable:Application` repräsentieren. Für die entsprechende Abbildung muss das MoNA-Attribut `type`, welches Auskunft über den Namen des Kommunikationsdienstes gibt, auf die Eigenschaft `uco-core:name` der `Application` übertragen werden. Tabelle 5.5 zeigt diese beschriebene Abbildung von MoNA auf CASE. Als UUID wird der Hashwert verwendet, den die `hashCode()`-Methode der entsprechenden MoNA-Instanz zurückgibt. Einem mobilen Kommunikationsdienst sind mehrere Chats zugeordnet, was anhand der Definition einer Beziehung zwischen den Entitäten realisiert werden kann. Zur Modellierung der Beziehung eignet sich die CASE-Klasse `uco-observable:ObservableRelationship`. Die semantische Repräsentation eines Chats wird in Unterabschnitt 5.1.6 beschrieben.

Tabelle 5.5: Abbildung der Attribute eines mobilen Kommunikationsdienstes von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
`type`, `ChatType`, 1	`uco-observable:ApplicationFacet`	`uco-core:name`, `xsd:string`, 1

5.1.6 Semantische Repräsentation von Chats in CASE

Für die semantische Repräsentation von Chats kann die MoNA-Klasse `Chat` auf die CASE-Klasse `uco-observable:MessageThread` abgebildet werden. Dabei können die Attribute für den Namen (`name`), den Alias (`alias`), das Erstellungsdatum (`createdAt`), den Status beziehungsweise die Bio (`status`) und den zugehörigen Kommunikationsdienst (`communicationService`) eines Chats aus MoNA direkt als Eigenschaften eines `MessageThread` in CASE übertragen werden. Für den Kommunikationsdienst wird dabei die Klasse `uco-observable:Application` verwendet. Durch Nutzung der Eigenschaft `uco-core:name` kann der Dienst namentlich spezifiziert werden. Um zu definieren, ob ein Chat ein Gruppenchat ist, kann anstelle von `uco-core:name` über die Eigenschaft `uco-observable:groupName` der Gruppenname in CASE angegeben werden. Wenn ein `MessageThread` diese Eigenschaft nicht besitzt, lässt sich auf einen 1-zu-1-Chat zwischen zwei Accounts schlussfolgern. Als UUID einer `MessageThread`-Instanz kann der Hashwert, welchen die `hashCode()`-Methode der abzubildenden MoNA-Instanz zurückgibt, verwendet werden. Die `id` einer Nachricht aus MoNA lässt sich in das `uco-observable:MessageThreadFacet` als Wert der Eigenschaft `uco-observable:identifier` übernehmen. Die Definition eines Chat- oder Gruppenbilds erfolgt über eine Instanz der CASE-Klasse `uco-observable:RasterPicture`, analog zu der Modellierung eines Profilbilds von einem Account, wie in 5.1.4 erläutert. Zusammenfassend führt Tabelle 5.6 die Abbildung der MoNA-Attribute eines Chats in CASE auf.

Der Ersteller des Chats und die Administratoren können jeweils über eine Instanz der Klasse `uco-observable:ObservableRelationship` definiert werden. Die Quellinstanzen sind hierbei die Instanzen der Klasse `uco-observable:ApplicationAccount`. Die Zielinstanz ist der entsprechende `MessageThread`. Über die Eigenschaft `uco-observable:kindOfRelationship` einer `ObservableRelationship` kann die Rolle der Accounts im Chat spezifiziert werden.

Ein Chat enthält eine Menge von Nachrichten in zeitlich sortierter Reihenfolge. Dazu wird in der Eigenschaft `uco-observable:messageThread` des `uco-observable:MessageThreadFacet` eine Instanz der Klasse `uco-types:Thread` definiert. Die `Thread`-Instanz besitzt die Eigenschaft `co:size`, über welche die Anzahl der enthaltenen Nachrichten angegeben wird. Alle Nachrichten-IDs eines Chats werden in der `Thread`-Eigenschaft `co:element` aufgeführt. Die Reihenfolge wird über die Eigenschaft `co:item` definiert. Deren Wert ist eine Liste von `uco-types:ThreadItem`-Instanzen. Über die Eigenschaften eines `ThreadItem` lässt sich wiederum die Nachricht (`co:itemContent`), ihre Position (`co:index`) im Chat und ihre darauffolgende Nachricht (`uco-types:threadNextItem`) spezifizieren. Darüber hinaus werden Chatmitglieder angegeben, indem ihre `ApplicationAccount`-IDs in der Eigenschaft `uco-observable:participant` des jeweiligen `MessageThreadFacet` aufgelistet werden. Die semantische Repräsentation einer Nachricht in CASE führt Unterabschnitt 5.1.7 aus.

Tabelle 5.6: Abbildung der Attribute eines Chats von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
id, String, 1	uco-observable: MessageThreadFacet	uco-observable:identifier, xsd:string, 1
name, String, 1	uco-observable: MessageThread	uco-core:name, xsd:string, 1
alias, String, 1	owl:Thing (abgeleitet)	uco-observable:displayName, xsd:string, 1
createdAt, Date, 1	uco-observable: MessageThread	uco-core:objectCreatedTime, xsd:dateTime, 1
group, boolean, 1	owl:Thing (abgeleitet)	uco-observable:groupName, xsd:string, 1
photoUrl, String, 1	uco-observable: RasterPicture	uco-observable:filePath xsd:string, 1
status, String, 1	uco-observable: MessageThread	uco-core:description, xsd:string, 1
communicationService, ChatType, 1	owl:Thing (abgeleitet)	uco-observable:application, uco-observable:Application, 1

5.1.7 Semantische Repräsentation von Nachrichten in CASE

Nachrichten werden in MoNA durch `Message`-Instanzen beschrieben, wobei sie in CASE auf Instanzen der Klasse `uco-observable:Message` abgebildet werden können. Als UUID einer entsprechenden CASE-Instanz kann dabei der Hashwert, welchen die Methode `hashCode()` für eine Nachricht in MoNA zurückgibt, verwendet werden. Die eigentliche Nachrichten-(`id`) der MoNA-Instanz wird als `uco-observable:messageID` des `uco-observable:MessageFacet` in CASE übernommen. Das `MessageFacet` stellt weitere Eigenschaften für eine Nachricht bereit, wobei unter anderem der Nachrichtentext (`message`) und die Sendezeit (`time`) aus MoNA direkt durch eine jeweilige Eigenschaft abgebildet werden kann. Zudem gibt die Eigenschaft `uco-observable:messageType` des `MessageFacet` den Nachrichtentyp an. Somit lässt sich angeben, ob in der Nachricht beispielsweise Text, Medien (Bild, Video, Audio, Dokument u.Ä.), Standorte oder Kontaktdaten versendet wurden oder Informationen über einen getätigten Anruf enthalten sind. Durch die Eigenschaften `uco-observable:from` und `uco-observable:to` der Klasse `uco-observable:MessageFacet` werden jeweils der Sender und die Empfänger der Nachricht spezifiziert. Hierzu müssen die entsprechenden `uco-observable:ApplicationAccount`-IDs angegeben werden. Falls eine Nachricht Anrufdaten enthält, lässt sich die Anrufdauer unter Verwendung des `uco-observable:CallFacet` über dessen Eigenschaft `uco-observable:duration` darstellen. Die Abbildung der MoNA-Attribute einer Nachricht auf CASE fasst Tabelle 5.7 zusammen.

Tabelle 5.7: Abbildung der Attribute einer Nachricht von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
id, String, 1	uco-observable:MessageFacet	uco-observable:messageID, xsd:string, 1
message, String, 1	uco-observable:MessageFacet	uco-observable:messageText, xsd:string, 1
time, long, 1	uco-observable:MessageFacet	uco-observable:sentTime, xsd:dateTime, 1
type, MessageType, 1	uco-observable:MessageFacet	uco-observable:messageType, xsd:string, 1
from, ContactId, 1	uco-observable:MessageFacet	uco-observable:from, uco-observable: ApplicationAccount, 1
to, Chat, 1	uco-observable:MessageFacet	uco-observable:to, uco-observable: ApplicationAccount, 0...n
duration, int, 1	uco-observable:CallFacet	uco-observable:duration, xsd:integer, 1

Die Datenquelle, beispielsweise eine Datenbank, kann über die Definition einer Instanz der Klasse `uco-observable:ObservableRelationship` einer Nachricht zugeordnet werden. Die Eigenschaft `uco-core:kindOfRelationship` hat in diesem Fall den Wert „`Contained_Within`". Als `uco-core:source` wird die ID der Nachricht angegeben, während `uco-core:target` der ID der Datenquelle entspricht. Die Datenquelle wird separat als Instanz der Klasse `uco-observable:File` in CASE definiert. Dabei stellt das `uco-observable:FileFacet` der Instanz Eigenschaften bereit. Somit lassen sich zum Beispiel Erstellungsdatum (`uco-observable:observableCreatedTime`), Dateiname (`uco-observable:fileName`) und Dateigröße (`uco-observable:sizeInBytes`) angeben.

Multimediadateien, welche als Nachricht in einem Chat versendet oder empfangen wurden, können über eine Instanz der Klasse `ObservableRelationship` mit der zugehörigen `Message`-Instanz in Beziehung gesetzt werden. Wie sich Mediendateien in CASE repräsentieren lassen, beschreibt Unterabschnitt 5.1.8.

5.1.8 Semantische Repräsentation von Multimediadateien in CASE

In MoNA werden Multimediadateien durch die Klasse `Media` beschrieben. Zur Abbildung in CASE eignet sich hierfür die Klasse `uco-observable:ContentData`, welche unter anderem Eigenschaften bezüglich des Erstellungsdatums (`uco-core:objectCreatedTime`) und des Änderungsdatums (`uco-core:modifiedTime`) bereitstellt. Als UUID einer derartigen CASE-Instanz kann hierbei der Hashwert verwendet werden, welchen das MD5-Verfahren für die Originaldatei zurückgibt. Der Pfad einer entsprechenden Mediendatei wird über die Eigenschaft `uco-observable:filePath` der Klasse `uco-observable:FileFacet` angegeben. Durch die Eigenschaft `uco-observable:mimeType` des `uco-observable:ContentDataFacet` kann außerdem der Typ und das Format der Mediendatei spezifiziert werden. Die Abbildung einer Multimediadatei von MoNA auf CASE wird in Tabelle 5.8 zusammenfassend dargestellt.

Tabelle 5.8: Abbildung der Attribute einer Multimediadatei von dem MoNA-Datenmodell auf die CASE-Ontologie.

MoNA-Attribut (Name, Typ, Kardinalität)	**CASE-Klasse**	**CASE-Eigenschaft (Name, Typ, Kardinalität)**
`localPath`, `String`, 1	`uco-observable:FileFacet`	`uco-observable:filePath`, `xsd:string`, 1
`timestampCreation`, `long`, 1	`uco-observable:ContentData`	`uco-core:objectCreatedTime`, `xsd:dateTime`, 1
`timestampModified`, `long`, 1	`uco-observable:ContentData`	`uco-core:modifiedTime`, `xsd:dateTime`, 1
`type`, `MediaType`, 1	`uco-observable:ContentDataFacet`	`uco-observable:mimeType`, `xsd:string`, 1

5.1.9 Semantische Repräsentation von nicht in CASE erfassten Daten

Die vorherigen Unterabschnitte dieses Kapitels beschreiben die generalisierte Abbildung mobiler Kommunikation von MoNA auf eine Subdomäne von CASE. Diese Abbildung kann für andere forensische Tools entsprechend adaptiert werden. Jedoch besitzt jede Software eigene Analysemethoden, deren Ergebnisse in der Regel ebenfalls gespeichert werden. Aktionen, die in einer Untersuchung durchgeführt wurden, können durch die Klasse `case-investigation:InvestigativeAction` dargestellt werden [32]. Allerdings kann der CASE-Standard die Repräsentation spezifischer Ergebnisse aus Tools nicht in jedem Fall abdecken, da CASE die Cyberdomäne im Bereich der forensischen Untersuchung generalisiert abbildet. Falls im Rahmen der Abbildung von Tool-Instanzen auf CASE Lücken identifiziert werden konnten, empfiehlt das CASE-Konsortium für entsprechende Klassen die Definition eigener `Facets` [33]. Wenn die Art der Daten ebenfalls nützlich für andere Anwender von CASE ist, kann dem Konsortium vorgeschlagen werden, sie in den CASE-Standard aufzunehmen.

Beispielsweise ermöglicht MoNA die Markierung von Instanzen als relevant oder irrelevant. Dabei kann die Markierung manuell vom Ermittler oder automatisch durch einen Service von MoNA vorgenommen werden. CASE stellt hierzu keine geeignete Klasse zur Repräsentation bereit, weshalb ein eigenes `Facet` definiert werden müsste. Quelltext 5.1 zeigt in diesem Fall beispielhaft wie ein derartiges `Facet` im RDF-Format definiert werden kann. Hierbei werden neben dem Label (Zeile 10) und dem Kommentar (Zeile 11-12) die zugehörigen Eigenschaften (Zeile 14-21) angegeben. In diesem Fall handelt es sich um zwei Eigenschaften, die jeweils einen booleschen Wert als Literal annehmen können. Eine Eigenschaft dient zur Repräsentation manueller Relevanz-Markierungen, während die andere Eigenschaft Informationen über automatisch gesetzte Relevanz-Markierungen abbildet. Die Eigenschaften selbst müssen darüber hinaus zusätzlich definiert werden, da in CASE noch keine geeignete `owl:DatatypeProperty` zur Referenzierung existiert. Die Definition der beiden Eigenschaften des `Facets` ist jeweils in Quelltext A.1 und Quelltext A.2 im Anhang A gegeben.

```
1  @prefix core: <https://ontology.unifiedcyberontology.org/uco/core/> .
2  @prefix analysis: <https://ontology.unifiedcyberontology.org/uco/analysis/> .
3  @prefix owl: <http://www.w3.org/2002/07/owl#> .
4  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
5  @prefix sh: <http://www.w3.org/ns/shacl#> .
6  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
7
8  analysis:MoNARelevanceFacet a owl:Class,
9          sh:NodeShape ;
10     rdfs:label "MoNARelevanceFacet"@en ;
11     rdfs:comment "A MoNA relevance facet is a grouping of characteristics
12         unique to represent relevance tags from the MoNA software."@en ;
13     rdfs:subClassOf core:Facet ;
14     sh:property [ sh:datatype xsd:boolean ;
15             sh:maxCount 1 ;
16             sh:nodeKind sh:Literal ;
17             sh:path analysis:isManuallyRelevant ],
18         [ sh:datatype xsd:boolean ;
19             sh:maxCount 1 ;
20             sh:nodeKind sh:Literal ;
21             sh:path analysis:isAutomaticallyRelevant ] ;
22     sh:targetClass analysis:MoNARelevanceFacet .
```

Quelltext 5.1: Definition eines Facets in RDF zur Repräsentation von Relevanz-Markierungen aus MoNA.

5.2 Implementierung und Anwendung des semantischen Webservers

Nach dem Aufsetzen des Apache Tomcat-Servers ist dieser über `http://localhost:8080/` für das virtualisierte Ubuntu-System verfügbar. Vom Windows-System kann der Server über `http://192.168.56.101:8080/` erreicht werden, wie in Abbildung 5.2 zu sehen ist. Unter Ubuntu lässt sich der Tomcat-Server verwalten, falls der Nutzer Administratorrechte besitzt. Insbesondere über die „Manager App“ von Tomcat ist das Erstellen, Konfigurieren und Entfernen von Webanwendungen möglich. Somit kann der Administrator in der Tomcat-Weboberfläche ebenfalls auf die Verfügbarkeit der Apache Jena Fuseki-Anwendung Einfluss nehmen.

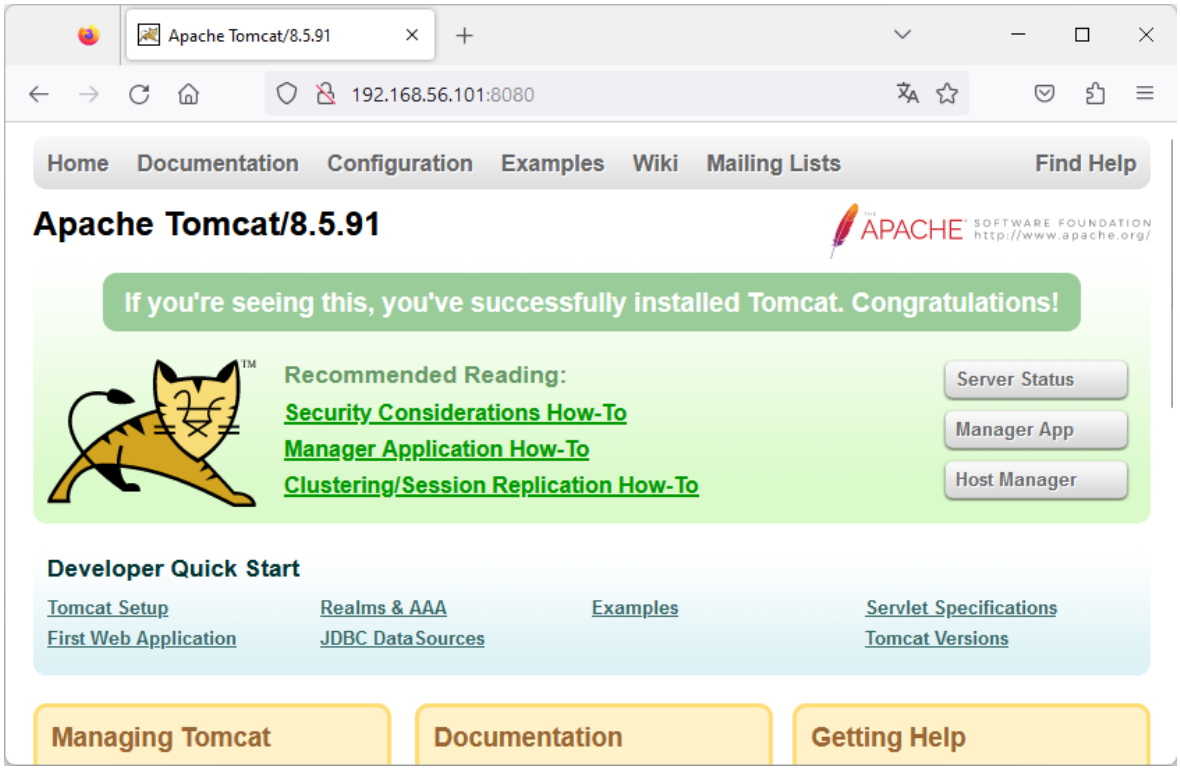

Abbildung 5.2: Weboberfläche des aufgesetzten Apache Tomcat-Servers.

Durch die Kombination des Apache Tomcat-Servers mit der Apache Jena Fuseki-Anwendung kann ein semantischer Webserver bereitgestellt werden. Dieser Server lässt sich auf dem Ubuntu-System unter `http://localhost:8080/fuseki/#/` aufrufen. Für den Windows-Host der virtualisierten Maschine ist der Fuseki-Server über `http://192.168.56.101:8080/fuseki/#/` erreichbar, wie Abbildung 5.3 zeigt. In der Weboberfläche von Fuseki können Datensätze durchsucht und verwaltet werden. Darüber ist die Dokumentation von Fuseki aufrufbar. Beim Erstellen eines Datensatzes wird der Nutzer gefragt, ob die Daten temporär im Speicher gehalten oder via TDB2 persistiert werden sollen. Im Anschluss lassen sich dem benannten Datensatz Daten hinzufügen. Hierzu akzeptiert Fuseki RDF-Formate wie Turtle, RDF/XML oder TRiG.

Des Weiteren ermöglicht Fuseki die Ausführung von Anfragen in SPARQL auf Datensätzen. Dabei muss je nach Art der Anfrage der richtige „SPARQL Endpoint“ ausgewählt werden. Der Endpoint beginnt immer mit `/<DATENSATZNAME>/`, wobei <DATENSATZNAME> durch den Namen des jeweiligen Datensatzes ersetzt werden muss. Zum Ausführen einer Query, beispielsweise eines `SELECT`-Statements, wird daraufhin das Suffix `query` oder alternativ `sparql` angefügt. Um Daten einzufügen,

zu manipulieren oder zu löschen, muss der Endpoint `/<DATENSATZNAME>/update` verwendet werden. Die Ergebnisse einer Anfrage können wahlweise in JSON, XML, CSV oder TSV zurückgegeben werden. Der semantische Webserver wurde im Rahmen dieser Arbeit zum Testen der entworfenen Ontologie verwendet, wie in Abschnitt 4.3 beschrieben. Die erzielten Ergebnisse sind in Abschnitt 5.3 zusammengefasst.

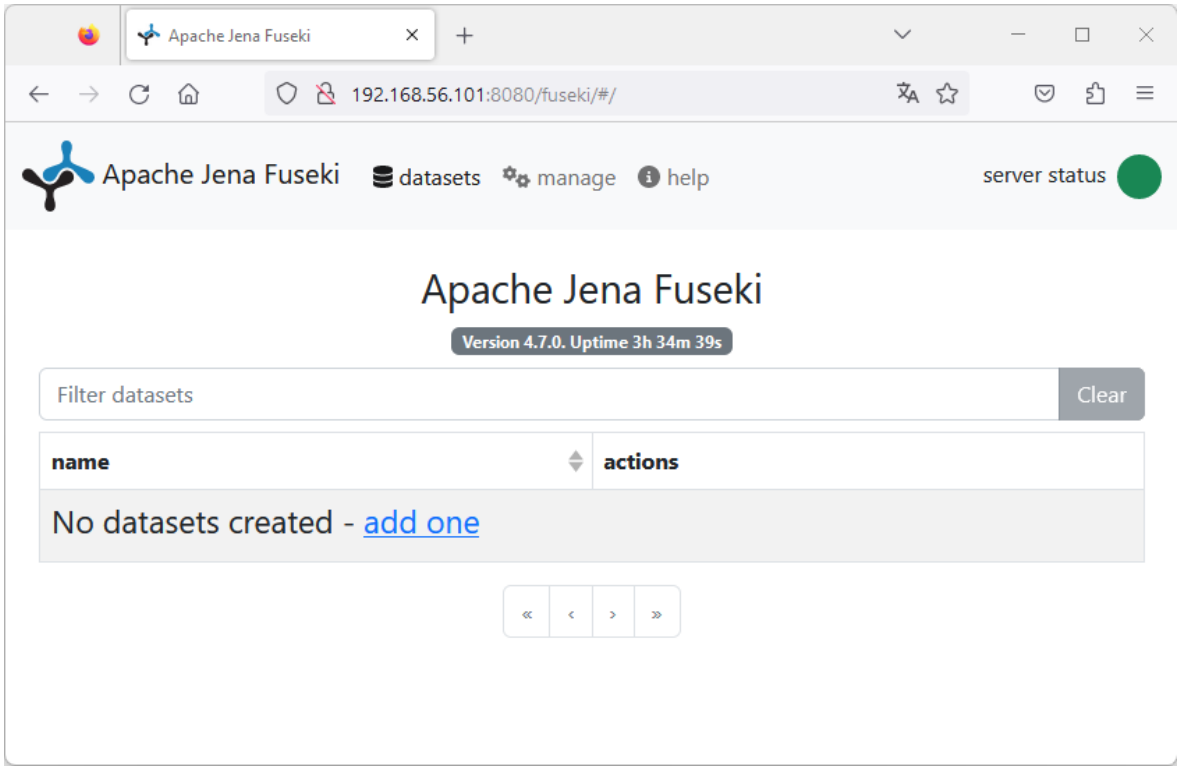

Abbildung 5.3: Weboberfläche des aufgesetzten Apache Jena Fuseki-Servers.

5.3 Testergebnisse der entworfenen Ontologie

Eine Ontologie zur Repräsentation mobiler Kommunikation konnte im JSON-LD-Format mithilfe der entwickelten Exportfunktion aus dem Datenmodell von MoNA erstellt werden. Dieser Abschnitt behandelt ausschnittweise die wichtigsten Instanzen der im Test erzeugten Ontologie. Die aufgeführten Beispiele verdeutlichen dabei, wie die in Abschnitt 5.1 beschriebene Abbildung auf CASE in der Praxis implementiert werden kann. Der Inhalt der exportierten JSON-LD-Datei beginnt mit der Definition des Kontexts, welcher alle Namensräume definiert. Im Anwendungstest wurde der in Quelltext 5.2 abgebildete Kontext erzeugt. Instanz-Daten, die aus MoNA stammen, werden in den folgenden Ausführungen durch das Präfix `http://mona.de/kb/`, abgekürzt mit `kb`, gekennzeichnet.

```
"@context": {
  "kb": "http://mona.de/kb/",
  "investigation": "https://ontology.caseontology.org/case/investigation/",
  "uco-core": "https://ontology.unifiedcyberontology.org/uco/core/",
  "uco-identity": "https://ontology.unifiedcyberontology.org/uco/identity/",
  "uco-observable":
    "https://ontology.unifiedcyberontology.org/uco/observable/",
  "uco-types": "https://ontology.unifiedcyberontology.org/uco/types/",
  "co": "http://purl.org/co/",
  "xsd": "http://www.w3.org/2001/XMLSchema#"
}
```

Quelltext 5.2: Kontext der im Anwendungstest erzeugten Ontologie.

Quelltext 5.3 führt die Definition einer Untersuchung auf. Hierbei folgt nach dem Kontext ein `Bundle` (Zeile 6-30), welches alle anderen Instanzen einer forensischen Untersuchung beinhaltet. Als erste Instanz ist eine Untersuchung und ihre Eigenschaften enthalten (Zeile 10-27). Die Untersuchung hat in diesem Fall die Auftragsnummer `12345` (Zeile 13) und den Namen `Test` (Zeile 14). Sie wurde am 16. Oktober 2023 um 20:08:45 Uhr erstellt (Zeile 15-18). Dabei sind zwei mobile Endgeräte im Fokus der Untersuchung (Zeile 19-26). Alle weiteren Instanzen werden im Quelltext ab Zeile 28 definiert und in den folgenden Quelltexten ausschnittweise gezeigt.

```
{
  "@context": {
    ...
  },
  "@graph": [
    {
      "@id": "kb:bundle-_DDUmEGxPEe6xUdLaH3gMNw",
      "@type": "uco-core:Bundle",
      "uco-core:object": [
        {
          "@id": "kb:investigation-_DDUmEGxPEe6xUdLaH3gMNw",
          "@type": "investigation:Investigation",
          "investigation:rootExhibitNumber": "12345",
          "uco-core:name": "Test",
          "uco-core:objectCreatedTime": {
            "@type": "xsd:dateTime",
            "@value": "2023-10-16T20:08:45"
          },
          "uco-core:object": [
```

```
            {
              "@id": "kb:mobile-device-4495775fa4f7f83ffab60cd614c1e2ff"
            },
            {
              "@id": "kb:mobile-device-2ac7c38f80a6a3dfd72a1c4cb1c19c75"
            }
          ]
        },
        ...
      ]
    }
  ]
}
```

Quelltext 5.3: Eine forensische Untersuchung ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Der Analyst und die ermittelnde Dienststelle wird in Quelltext 5.4 instanziiert. In diesem Test ist der Analyst `Lukas Jaeckel` (Zeile 2-6) und die Dienstelle die `Hochschule Mittweida` (Zeile 7-11). Daraufhin wird im Quelltext jeweils die Beziehung der Instanzen zu der forensischen Untersuchung definiert (Zeile 12-35).

```
...
  {
    "@id": "kb:examiner-8d7c1ad609503b9053bdccac50bdcd49",
    "@type": "investigation:Examiner",
    "uco-core:name": "Lukas Jaeckel"
  },
  {
    "@id": "kb:organization-f6bad830a799bca759e18b51b2768f6c",
    "@type": "identity:Organization",
    "uco-core:name": "Hochschule Mittweida"
  },
  {
    "@id": "kb:relationship-2f7b530a98e1544e00358449219a1db9",
    "@type": "uco-core:ObservableRelationship",
    "uco-core:source": {
      "@id": "kb:investigation-_DDUmEGxPEe6xUdLaH3gMNw"
    },
    "uco-core:target": {
      "@id": "kb:examiner-8d7c1ad609503b9053bdccac50bdcd49"
    },
    "uco-core:kindOfRelationship": "Has_Examiner",
    "uco-core:isDirectional": true
  },
  {
    "@id": "kb:relationship-0bb5b14b49b9b4913ccc13e48c05bbce9bcfea33",
    "@type": "uco-core:ObservableRelationship",
    "uco-core:source": {
      "@id": "kb:investigation-_DDUmEGxPEe6xUdLaH3gMNw"
    },
    "uco-core:target": {
      "@id": "kb:organization-f6bad830a799bca759e18b51b2768f6c"
    },
```

```
33      "uco-core:kindOfRelationship": "Has_Department",
34      "uco-core:isDirectional": true
35    },
36  ...
```

Quelltext 5.4: Ein forenischer Auswerter und eine Dienststelle ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Die Instanziierung eines der beiden mobilen Endgeräte zeigt ausschnittweise Quelltext 5.5 in Zeile 7-32. Zuvor wird eine Instanz der Gerätemarke `Samsung` erstellt (Zeile 2-6). Diese Instanz wird im `DeviceFacet` des Geräts in den Zeilen 20-22 referenziert. Für das Gerät wurde der Name `Galaxy S9 SM-G960F von Johannes` vergeben (Zeile 10). Es wurde am 16.10.2023 um 20:10:25 Uhr eingelesen (Zeile 11-14) und besitzt die Asservatennummer `001` (Zeile 15). Die Modellbezeichnung des Geräts lautet `Galaxy S9 SM-G960F` (Zeile 23).

```
1  ...
2    {
3      "@id": "kb:organization-3910b1e0ccab19bc46fd9db27cca49c9",
4      "@type": "uco-identity:Organization",
5      "uco-core:name": "Samsung"
6    },
7    {
8      "@id": "kb:mobile-device-4495775fa4f7f83ffab60cd614c1e2ff",
9      "@type": "uco-observable:MobileDevice",
10     "uco-core:name": "Galaxy S9 SM-G960F von Johannes",
11     "uco-observable:objectCreatedTime": {
12       "@type": "xsd:dateTime",
13       "@value": "2023-10-16T20:10:25"
14     },
15     "investigation:exhibitNumber": "001"
16     "uco-core:hasFacet": [
17       {
18         "@id": "kb:device-facet-4495775fa4f7f83ffab60cd614c1e2ff",
19         "@type": "uco-observable:DeviceFacet",
20         "uco-observable:manufacturer": {
21           "@id": "kb:organization-3910b1e0ccab19bc46fd9db27cca49c9"
22         },
23         "uco-observable:model": "Galaxy S9 SM-G960F"
24       },
25       {
26         "@id": "kb:operating-system-facet-4495775fa4f7f83ffab60cd614c1e2ff",
27         "@type": "uco-observable:OperatingSystemFacet",
28         "uco-core:name": "Android",
29         "uco-observable:version": "10"
30       }
31     ]
32   },
33 ...
```

Quelltext 5.5: Ein mobiles Endgerät ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Quelltext 5.6 zeigt exemplarisch einen Ausschnitt, welcher eine Person repräsentiert (Zeile 2-6), aus der JSON-LD-Datei. Hierbei ist dieser Person der Name `Johannes` zugeordnet (Zeile 5). Zudem ist eine Beziehung zwischen der Person und dem in Quelltext 5.5 abgebildeten mobilen Endgerät definiert, welche die Person als Eigentümer des Geräts charakterisiert (Zeile 7-18).

```
1  ...
2    {
3      "@id": "kb:person-1268343946",
4      "@type": "uco-identity:Person",
5      "uco-core:name": "Johannes"
6    },
7    {
8      "@id": "kb:relationship-88c357d57c3b16e9f1de95120d37f19b",
9      "@type": "uco-core:ObservableRelationship",
10     "uco-core:source": {
11       "@id": "kb:mobile-device-4495775fa4f7f83ffab60cd614c1e2ff"
12     },
13     "uco-core:target": {
14       "@id": "kb:person-1268343946"
15     },
16     "uco-core:kindOfRelationship": "Has_Owner",
17     "uco-core:isDirectional": true
18   },
19 ...
```

Quelltext 5.6: Eine Person, die zugleich Geräteeigentümer ist, ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Die im Anwendungstest erzeugte Ontologie enthält für jedes Gerät jeweils den Messenger-Dienst WhatsApp, Telegram und Facebook Messenger. Quelltext 5.7 bildet hierbei ausschnittweise die WhatsApp-Anwendung aus der exportierten JSON-LD-Datei ab. Mithilfe des `ApplicationFacet` wird der Name des Kommunikationsdienst angegeben (Zeile 9). Andere Instanzen nutzten die in Zeile 3 festgelegte ID zur Referenzierung auf den WhatsApp-Dienst.

```
1  ...
2    {
3      "@id": "kb:application-1408950124",
4      "@type": "uco-observable:Application",
5      "uco-core:hasFacet": [
6        {
7          "@id": "kb:application-facet-1408950124",
8          "@type": "uco-observable:ApplicationFacet",
9          "uco-core:name": "WhatsApp"
10       }
11     ]
12   },
13 ...
```

Quelltext 5.7: Ein mobiler Kommunikationsdienst ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Quelltext 5.8 führt ausschnittweise auf, wie ein Account in der exportierten JSON-LD-Datei instanziiert wird. Dem Account ist der Name `Johannes` (Zeile 5) und die in Quelltext 5.7 gezeigte WhatsApp-Anwendung (Zeile 6-8) zugeordnet. Darüber hinaus hat der Account eine Telefonnummer (Zeile 9) und über das `AccountFacet` eine WhatsApp-ID als Account-Identifikator (Zeile 14). Anhand der in den Zeilen 18-29 definierten Beziehung wird der WhatsApp-Account der Person zugeordnet, deren Instanziierung Quelltext 5.6 zeigt.

```
1  ...
2    {
3      "@id": "kb:application-account-507299682",
4      "@type": "uco-observable:ApplicationAccount",
5      "uco-core:name": "Johannes",
6      "uco-observable:application": {
7        "@id": kb:application-1408950124
8      },
9      "uco-observable:phoneNumber": "+491724612271",
10     "uco-core:hasFacet": [
11       {
12         "@id": "kb:account-facet-507299682",
13         "@type": "uco-observable:AccountFacet",
14         "uco-observable:accountIdentifier": "491724612271@s.whatsapp.net"
15       }
16     ]
17   },
18   {
19     "@id": "kb:relationship-54871ee73f37ac11dde40a0c8ea00ccc",
20     "@type": "uco-core:ObservableRelationship",
21     "uco-core:source": {
22       "@id": "kb:person-1268343946"
23     },
24     "uco-core:target": {
25       "@id": "kb:application-account-507299682"
26     },
27     "uco-core:kindOfRelationship": "Has_Application_Account",
28     "uco-core:isDirectional": true
29   },
30 ...
```

Quelltext 5.8: Ein Account ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Ein Profilbild eines Accounts kann als Instanz der Klasse `uco-observable:RasterPicture` repräsentiert werden. Hierbei zeigt Quelltext 5.9 exemplarisch einen Ausschnitt, in dem ein derartiges Bild instanziiert ist (Zeile 2-7). Dazu wird in Zeile 6 dessen relativer Dateipfad angegeben. Die Zuordnung des Bilds zu einem Account mit der ID `kb:application-account-923794548` erfolgt über die Definition einer Beziehung (Zeile 8-19).

```
...
  {
    "@id": "kb:raster-picture-9a0d5cc52eb7e06460a78a845b1a5203",
    "@type": "uco-observable:RasterPicture",
    "uco-observable:filePath:
      "\media\telegram-peer-photo-size-2-5380108448991130369-1-0-0",
  },
  {
    "@id": "kb:relationship-01d6e9622cfc681fc2d91a14c734285a",
    "@type": "uco-core:ObservableRelationship",
    "uco-core:source": {
      "@id": "kb:application-account-923794548"
    },
    "uco-core:target": {
      "@id": "kb:raster-picture-9a0d5cc52eb7e06460a78a845b1a5203"
    },
    "uco-core:kindOfRelationship": "Has_Profile_Picture",
    "uco-core:isDirectional": true
  },
...
```

Quelltext 5.9: Das Profilbild eines Accounts ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Der in Quelltext 5.10 aufgeführte Ausschnitt aus der exportierten JSON-LD-Datei zeigt die Repräsentation eines Chats. Bei diesem Chat handelt es sich um einen Gruppenchat mit dem Namen `SOS` (Zeile 5). Die Gruppe wurde am 01.10.2021 um 16:01:49 Uhr erstellt (Zeile 6-9) und stammt aus der in Quelltext 5.7 definierten WhatsApp-Anwendung (Zeile 10-12). Dabei ist der Gruppe intern in WhatsApp die ID `4915258469455-1633096909@g.us` zugeordnet (Zeile 17). Insgesamt wurden in der Gruppe `122` Nachrichten ausgetauscht (Zeile 21-24), wobei die ID jeder Nachricht als Eigenschaft `co:element` aufgeführt wird (Zeile 25-31). Der Quelltext ist hierbei für eine bessere Darstellung gekürzt. Danach folgt die Eigenschaft `co:element`, welche die Reihenfolge der Nachrichten definiert (Zeile 32-50). In dem gekürzten Ausschnitt ist dabei die Nachricht mit der ID `kb:message-1042861605` die in zeitlich aufsteigender Reihenfolge 35. Nachricht im Gruppenchat (Zeile 34-43). Auf ihr folgt die Nachricht, welche dem `uco-types:ThreadItem` mit der ID `kb:thread-item-1062177717` zugeordnet ist (Zeile 44-46). Abschließend werden die Account-IDs aller Gruppenmitglieder im Quelltext verkürzt aufgeführt (Zeile 51-56). Darunter befindet sich auch die ID des in Quelltext 5.8 gezeigten Accounts (Zeile 52-54).

```
...
  {
    "@id": "kb:thread-1570092021",
    "@type": "uco-observable:MessageThread",
    "uco-observable:groupName": "SOS",
    "uco-core:objectCreatedTime": {
      "@type": "xsd:dateTime",
      "@value": "2021-10-01T16:01:49"
    },
    "uco-observable:application": {
      "@id": kb:application-1408950124
    },
    "uco-core:hasFacet": [
      {
```

```
        "@id": "kb:message-thread-facet-1570092021",
        "@type": "uco-observable:MessageThreadFacet",
        "identifier": "4915258469455-1633096909@g.us",
        "uco-observable:messageThread": {
          "@id": "kb:thread-1570092021",
          "@type": "uco-types:Thread",
          "co:size": {
            "@type": "xsd:nonNegativeInteger",
            "@value": "122"
          },
          "co:element": [
            ...
            {
              "@id": "kb:message-1042861605"
            },
            ...
          ],
          "co:item": [
            ...
            {
              "@id": "kb:thread-item-1042861605",
              "@type": "uco-types:ThreadItem",
              "co:index": {
                "@type": "xsd:positiveInteger",
                "@value": "35"
              },
              "co:itemContent": {
                "@id": "kb:message-1042861605"
              },
              "uco-types:threadNextItem": {
                "@id": "kb:thread-item-1062177717"
              }
            },
            ...
          ]
        },
        "uco-observable:participant": [
          {
            "@id": "kb:account-507299682"
          },
          ...
        ]
      }
    ]
  },
...
```

Quelltext 5.10: Ein Chat ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Quelltext 5.11 zeigt ausschnittweise die Repräsentation der 35. Nachricht des Gruppenchats, den Quelltext 5.10 abbildet. In MoNA ist der Nachricht die ID `8257BE31B6460DBC2D5816024E8E68C7` zugeordnet (Zeile 9). Dabei handelt es sich um eine ausgehende Textnachricht (Zeile 10), die über die WhatsApp-Anwendung aus Quelltext 5.7 versendet wurde (Zeile 11-13). Der Textinhalt `Not gonna lie, you should go to the police ASAP` (Zeile 14-15) wurde vom in Quelltext 5.8 aufgeführten Account verschickt (Zeile 16-18). Die Nachricht erhielten alle Accounts aus der Gruppe, deren IDs im Quelltext zwischen Zeile 19 und 24 angegeben sind. Zur besseren Darstellung ist der Quelltext hierbei verkürzt und zeigt nur von einem Empfänger die Account-ID in Zeile 21. Die Nachricht wurde am 01.10.2021 um 16:25:31 Uhr versendet (Zeile 25-28).

```
1  ...
2    {
3      "@id": "kb:message-1042861605",
4      "@type": "uco-observable:Message",
5      "uco-core:hasFacet": [
6        {
7          "@id": "kb:message-facet-1042861605",
8          "@type": "uco-observable:MessageFacet",
9          "uco-observable:messageID": "8257BE31B6460DBC2D5816024E8E68C7",
10         "uco-observable:messageType": "Outgoing text",
11         "uco-observable:application": {
12           "@id": "kb:application-1408950124"
13         },
14         "uco-observable:messageText":
15           "Not gonna lie, you should go to the police ASAP",
16         "uco-observable:from": {
17           "@id": "kb:account-507299682"
18         },
19         "uco-observable:to": [
20           {
21             "@id": "kb:account-873971803"
22           },
23           ...
24         ],
25         "uco-observable:sentTime": {
26           "@type": "xsd:dateTime",
27           "@value": "2021-10-01T16:25:31"
28         }
29       }
30     ]
31   },
32 ...
```

Quelltext 5.11: Eine Nachricht ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Um die Datenquelle einer Nachricht spezifizieren zu können, muss zuerst eine Repräsentation für die Quelle instanziiert werden. Hierbei eignet sich die Definition einer Datei, wie in Quelltext 5.12 ausschnittweise aus der exportierten JSON-LD-Datei aufgeführt wird (Zeile 2-14). In diesem Fall besitzt die Datenquelle die Dateiendung `db` (Zeile 9). Die Datei kann unter dem relativen Pfad `\com.whatsapp\databases\msgstore.db` vorgefunden werden (Zeile 10). Zusätzlich ist in Zeile 11 angegeben, dass es sich bei der Datei um kein Verzeichnis handelt. Demzufolge ist die Datenquelle eine Datenbankdatei. Die Verknüpfung von der Nachricht und ihrer Quelle erfolgt über die Beziehung, welche im Quelltext in den Zeilen 15-26 definiert ist.

```
1  ...
2    {
3      "@id": "kb:file-389e8ca75aaf8994895c65e1c3e357e6",
4      "@type": "uco-observable:File",
5      "uco-core:hasFacet": [
6        {
7          "@id": "kb:file-facet-389e8ca75aaf8994895c65e1c3e357e6",
8          "@type": "uco-observable:FileFacet",
9          "uco-observable:extension": "db",
10         "uco-observable:fileName": "\com.whatsapp\databases\msgstore.db",
11         "uco-observable:isDirectory": false
12       }
13     ]
14   },
15   {
16     "@id": "kb:relationship-4ddf3479d06298bf6262f3ad8331d763",
17     "@type": "uco-observable:ObservableRelationship",
18     "uco-core:source": {
19       "@id": "kb:message-1042861605"
20     },
21     "uco-core:target": {
22       "@id": "kb:file-389e8ca75aaf8994895c65e1c3e357e6"
23     },
24     "uco-core:kindOfRelationship": "Contained_Within",
25     "uco-core:isDirectional": true
26   },
27 ...
```

Quelltext 5.12: Die Datenquelle einer Nachricht ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Quelltext 5.13 zeigt exemplarisch anhand eines Ausschnitts aus der exportierten Ontologie im JSON-LD-Formt, wie eine Multimediadatei repräsentiert wird (Zeile 2-18). In diesem Fall handelt es sich um eine Bilddatei im JPEG-Format (Zeile 9). Dabei ist für sie der relative Pfad `Media/WhatsApp Images/IMG-20211001-WA0000.jpg` angegeben (Zeile 15). Durch die Definition einer Beziehung kann die Multimediadatei entweder als lokale Mediendatei mit einem mobilen Endgerät oder als Anhang mit einer Nachricht verknüpft werden. In diesem Beispiel wurde die Datei als Anhang der Nachricht mit der ID `kb:message-1284069113` versendet, was die Beziehung in den Zeilen 19-30 repräsentiert.

```
...
  {
    "@id": "kb:attachment-file-6c7f9a3a12a220e95518734f96909ba70da0a916",
    "@type": "uco-observable:ContentData",
    "uco-core:hasFacet": [
      {
        "@id": "kb:content-data-facet-6
   c7f9a3a12a220e95518734f96909ba70da0a916",
        "@type": "uco-observable:ContentDataFacet",
        "uco-observable:mimeType": "image/jpeg"
      },
      {
        "@id": "kb:file-facet-6c7f9a3a12a220e95518734f96909ba70da0a916",
        "@type": "uco-observable:FileFacet",
        "uco-observable:filePath":
          "Media/WhatsApp Images/IMG-20211001-WA0000.jpg"
      }
    ]
  },
  {
    "@id": "kb:relationship-db46f94353d0e5ad4fdd0bf7d59b623c",
    "@type": "uco-observable:ObservableRelationship",
    "uco-core:source": {
      "@id": "kb:attachment-file-6c7f9a3a12a220e95518734f96909ba70da0a916"
    },
    "uco-core:target": {
      "@id": "kb:message-1284069113"
    },
    "uco-core:kindOfRelationship": "Attachment_Of",
    "uco-core:isDirectional": true
  },
...
```

Quelltext 5.13: Eine versendete Multimediadatei ausschnittweise aus der im Anwendungstest exportierten Ontologie im JSON-LD-Format.

Nachdem die Ontologie im JSON-LD-Format aus MoNA exportiert war, konnte sie über das Python-Skript aus Abschnitt 4.3 erfolgreich in das Turtle-Format konvertiert werden. Im Anschluss wurde ein neuer Datensatz auf dem semantischen Webserver aus Abschnitt 5.2 erstellt und die Daten der erzeugten Ontologie konnten hinzugefügt werden. Daraufhin ließen sich Anfragen über SPARQL stellen, um zu überprüfen, ob der Datensatz alle Daten enthält, die aus MoNA exportiert werden sollten. Die Query in Quelltext 5.14 fasst die durchgeführten Anfragen bezüglich der Anzahl von Kommunikationsdiensten, Kontakten, Gruppen, Nachrichten und Multimediadateien zusammen.

```
PREFIX uco-observable:
        <https://ontology.unifiedcyberontology.org/uco/observable/>

SELECT (COUNT(?app) AS ?Kommunikationsdienste) (COUNT(?account) AS ?Kontakte)
        (COUNT(?group) AS ?Gruppen) (COUNT(?message) AS ?Nachrichten)
        (COUNT(?media) AS ?Medien) (COUNT(?image) AS ?Bilder)
        (COUNT(?video) AS ?Videos) (COUNT(?audio) AS ?Audios)
        (COUNT(?document) AS ?Dokumente)
WHERE {
  {
    ?app a uco-observable:Application .
  }
```

```
  UNION
  {
    ?account a uco-observable:ApplicationAccount .
  }
  UNION
  {
    ?group a uco-observable:MessageThread .
    FILTER EXISTS {?group uco-observable:groupName ?o }
  }
  UNION
  {
    ?message a uco-observable:Message .
  }
  UNION
  {
    ?media a uco-observable:ContentData .
  }
  UNION
  {
    ?image a uco-observable:ContentDataFacet ;
            uco-observable:mimeType ?imageValue .
    FILTER CONTAINS (?imageValue, "image/")
  }
  UNION
  {
    ?video a uco-observable:ContentDataFacet ;
            uco-observable:mimeType ?videoValue .
    FILTER CONTAINS (?videoValue, "video/")
  }
  UNION
  {
    ?audio a uco-observable:ContentDataFacet ;
            uco-observable:mimeType ?audioValue .
    FILTER CONTAINS (?audioValue, "audio/")
  }
  UNION
  {
    ?document a uco-observable:ContentDataFacet ;
              uco-observable:mimeType ?documentValue .
    FILTER ( CONTAINS (?documentValue, "application/")
              || CONTAINS (?documentValue, "text/") )
  }
}
```

Quelltext 5.14: SPARQL-Anfrage zur Ermittlung der jeweiligen Anzahl von Kommunikationsdiensten, Kontakten, Gruppen, Nachrichten und Multimediadateien (Bilder, Videos, Audios, Dokumente) in einem Datensatz.

Nach Ausführung der in Quelltext 5.14 aufgeführten Query konnte ihr Ergebnis mit der Statistik in Tabelle 4.1 aus Abschnitt 4.3 verglichen werden, wobei eine Übereinstimmung in allen Kategorien festgestellt wurde. Demzufolge enthält die exportierte Ontologie Repräsentationen aller MoNA-Instanzen, die im Rahmen dieser Arbeit als relevant eingestuft wurden. Die Ontologie ist dementsprechend geeignet, um alle Klassen, welche essentiell zur Beschreibung mobiler Kommunikation sind, aus MoNA abzubilden. Durch die erfolgte Generalisierung der MoNA-Klassen und ihrer jeweiligen Attribute ist die beschriebene Vorgehensweise und Abbildung für andere forensische Tools direkt übertragbar.

6 Zusammenfassung und Ausblick

Im Rahmen forensischer Untersuchungen stellt die Auswertung mobiler Endgeräte Strafverfolgungsbehörden vor eine wachsende Anzahl von Herausforderungen. Insbesondere die Analyse mobiler Kommunikation gestaltet sich als zunehmend schwieriger. Beispielsweise verwenden Kommunikationsdienste proprietäre Datenformate und immer häufiger Verschlüsselungsverfahren. Darüber hinaus können Kriminelle ihre Daten teilweise oder vollständig von ihren Geräten löschen und ihre Kommunikation über mehrere Anwendungen aufteilen, um eine forensische Auswertung bewusst zu erschweren. Zudem stellt allein die immense Menge an Daten Ermittlungsbehörden vor große Probleme bezüglich des Aufwands von Zeit und Ressourcen. Zur Extraktion und Analyse von Daten mobiler Endgeräte existieren eine Vielzahl forensischer Tools, welche jeweils Stärken und Grenzen aufweisen. Um den wachsenden Herausforderungen in der digitalen Forensik begegnen zu können, wird es jedoch in Zukunft notwendig sein, teil- und vollautomatisierte Lösungen zu entwickeln und im Ermittlungsprozess zu integrieren. Um hierbei unter anderem die Stärken von forensischen Tools miteinander kombinieren zu können, bedarf es dazu der Schaffung eines standardisierten Austauschformats.

Als Lösungsansatz und zur Optimierung des Ermittlungsprozesses diskutiert die Wissenschaft im Bereich der digitalen Forensik hierzu die Verwendung von Ontologien. Eine Ontologie kann dabei das Wissen aus einer Domäne bezüglich Cyberuntersuchungen repräsentieren. Somit kann insbesondere die Interoperabilität zwischen Tools und Systemen realisiert werden. Des Weiteren verbessern Ontologien die Möglichkeiten für intelligente Analysen. Beispielsweise lassen sich logische Inferenzen durchführen, womit neues Wissen erlangt werden kann. Ebenso können Informationen aus verschiedenen Bereichen der Cyberdomäne durch Ontologien kombiniert und damit Zusammenhänge erkannt werden. Die diskutierte Literatur stellt mehrere Entwürfe und Proof of Concepts von Ontologien für die genannten Zwecke vor. Jedoch konnte sich bisher noch keine der Ontologien als Standard in der Praxis durchsetzen. Hierbei fehlt es an der breiten Akzeptanz und Anwendung von Seiten der Behörden sowie der Entwickler forensischer Tools.

Aktuell arbeitet ein Konsortium aus Forschungsinstituten, staatlichen Einrichtungen sowie Unternehmen kontinuierlich an der Weiterentwicklung von CASE. Diese Ontologie dient zur Repräsentation der Domäne von digitalen Untersuchungen. CASE erweitert hierbei die UCO, welche die übergeordnete Domäne der Cybersicherheit abbildet. Aufgrund der namhaften und breiten Zusammenstellung sowie der regelmäßigen Aktivitäten des Konsortiums wurde CASE als Grundlage für die Ontologie gewählt, welche im Rahmen dieser Arbeit zur Repräsentation mobiler Kommunikation im Kontext forensischer Untersuchungen entwickelt werden sollte. Hierbei orientierte sich diese Arbeit an einem allgemeinen Leitfaden und Beispielen des CASE-Konsortiums. Die Herausforderung bestand insbesondere in der Analyse der CASE-Ontologie. Dabei wurden relevante Klassen, Eigenschaften und Beziehungen zur Modellierung einer Subdomäne bezüglich mobiler Kommunikation identifiziert und sinnvoll miteinander in Korrelation gesetzt. Die entworfene Ontologie umfasst hauptsächlich Klassen zur Darstellung von forensischen Untersuchungen, mobilen Endgeräten, Personen, Accounts, Kommunikationsdiensten, Chats, Nachrichten und Multimediadateien. Neben der Repräsentation von Instanzen dieser Klassen ermöglicht die gebildete Subdomäne von CASE ebenfalls die Abbil-

dung von Aktionen, die in einer forensischen Untersuchung durchgeführt wurden. Falls die Ontologie jedoch Daten, wie spezifische Ergebnisse von forensischen Tools, nicht abdeckt, müssen eigene Klassen und Eigenschaften definiert werden, wie in dieser Arbeit beispielhaft gezeigt wurde.

Für einen Proof of Concept wurde das Datenmodell von MoNA, einer forensischen Analyseplattform für mobile Kommunikation, auf die entworfene Ontologie abgebildet. Dabei wurde eine Exportfunktion in MoNA entwickelt, die eingelesene Kommunikationsdaten in Form einer Ontologie zurückgibt. Die grundlegenden Informationen, welche die eingelesenen Daten enthielten, konnten hierbei direkt auf Instanzen der entworfenen Ontologie übertragen werden. Somit ist die Repräsentation mobiler Kommunikation im Rahmen einer forensischen Untersuchung generell möglich. Die Abbildung und beschriebene Methodik kann analog für andere Tools adaptiert werden. Um spezifische Ergebnisse von forensischen Tools abbilden zu können, müssen möglicherweise eigene Klassen und Eigenschaften definiert werden. Damit mithilfe einer Ontologie die Ergebnisse von Tools untereinander ausgetauscht werden können, müssen demzufolge selbstdefinierte Klassen und Eigenschaften von den Entwicklern dokumentiert und von den anderen Tools ausgewertet werden. Darüber hinaus bedarf es gegebenenfalls eines eigenen Exports und Imports von Dateien, die in der Ontologie referenziert sind. Des Weiteren muss in zukünftigen Implementierungen bedacht werden, ob und wie Informationen mit redundanten oder sich überschneidenden Inhalten in der Ontologie zusammengeführt werden sollen.

Für einen Anwendungstest mit generierten Kommunikationsdaten wurde zudem prototypisch ein semantischer Webserver unter Verwendung von Apache Tomcat und Apache Jena Fuseki auf einem virtualisierten Ubuntu-System aufgesetzt. Auf diesen Server können Ontologien geladen, verwaltet und mittels der Sprache SPARQL abgefragt werden. Im Test ließ sich somit erfolgreich überprüfen, dass die von der MoNA-Exportfunktion erzeugte Ontologie alle Daten, welche im Rahmen dieser Arbeit als relevant eingestuft wurden, aus dem entsprechenden MoNA-Datenmodell repräsentierte. Mittels Apache Jena[28] könnten Programme in zünftigen Implementierungen direkt auf den Server zugreifen und SPARQL-Anfragen stellen, wodurch eine neue Art der Suche ermöglicht wird. Zudem könnten regelbasierte oder ontologische Reasoner, wie zum Beispiel Pellet [67] oder HermiT [68], verwendet werden, was das logische Schlussfolgern auf neues Wissen ermöglichen würde. Im Praxiseinsatz müsste der semantische Webserver zusätzlich konfiguriert werden, um unter anderem eine Rechte- und Zugriffsverwaltung sowie eine Nutzerauthentifizierung zu implementieren. Darüber hinaus könnte der Server theoretisch als Backend für forensische Tools fungieren. Die Tools könnten in diesem Fall somit auf dieselbe Ontologie als Datenquelle zugreifen, während der Server für das Persistieren und die Synchronisierung von Daten sowie für die Zugriffsverwaltung zuständig wäre. Zur Umsetzung des semantischen Webservers als Backend stehen allerdings Performance-Tests in der Praxis aus.

Zusammengefasst zeigt diese Arbeit, wie mobile Kommunikation im Kontext forensischer Untersuchungen durch eine Ontologie repräsentiert werden kann und welche Chancen durch den Einsatz dieser Ontologie ermöglicht werden. In der Praxis muss sie allerdings sowohl von Entwicklern forensischer Tools als auch von Sachbearbeitern in Strafverfolgungsbehörden akzeptiert und genutzt werden, um ihr volles Potenzial entfalten zu können. Für diesen Zweck könnte in Zukunft das Ergebnis dieser Arbeit mit den Ontologie-Experten aus dem CASE-Konsortium ausgetauscht und diskutiert werden. Daraufhin muss je nach Ergebnis der Sozialisierung die entworfene Ontologie

[28] Website von Apache Jena: `https://jena.apache.org/`, besucht am 15.10.2023

gegebenenfalls angepasst oder erweitert werden. Da sich die Technologien bezüglich der mobilen Kommunikation stetig weiterentwickeln, können sich die Anforderungen an die Repräsentation von entsprechenden Informationen in Zukunft ändern. Demzufolge besteht die Herausforderung, dass die entwickelte Ontologie fortlaufend gepflegt werden muss. Zukünftige Arbeiten können darüber hinaus die Einsatzmöglichkeiten künstlicher Intelligenz unter Verwendung dieser Ontologie erforschen und diskutieren. Dazu lassen sich insbesondere Szenarien untersuchen, in denen durch logische Inferenz auf forensisch relevantes Wissen geschlossen werden kann.

Anhang A: Definition von Eigenschaften in RDF

```
@prefix analysis: <https://ontology.unifiedcyberontology.org/uco/analysis/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

analysis:isManuallyRelevant a owl:DatatypeProperty ;
    rdfs:label "isManuallyRelevant"@en ;
    rdfs:comment "Indicates whether the associated instance is tagged as
        manually relevant in MoNA"@en ;
    rdfs:range xsd:boolean .
```

Quelltext A.1: Definition einer Eigenschaft im RDF-Format zur Repräsentation von in MoNA manuell gesetzten Relevanz-Markierungen.

```
@prefix analysis: <https://ontology.unifiedcyberontology.org/uco/analysis/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

analysis:isAutomaticallyRelevant a owl:DatatypeProperty ;
    rdfs:label "isAutomaticallyRelevant"@en ;
    rdfs:comment "Indicates whether the associated instance is tagged as
        automatically relevant in MoNA"@en ;
    rdfs:range xsd:boolean .
```

Quelltext A.2: Definition einer Eigenschaft im RDF-Format zur Repräsentation von in MoNA automatisch gesetzten Relevanz-Markierungen.

Literaturverzeichnis

[1] eMarketer. „Anzahl der Nutzer von Messenger-Apps weltweit in den Jahren 2018 bis 2020 sowie eine Prognose bis 2025 (in Milliarden)“, Statista. (13. Sep. 2021), Adresse: https://de.statista.com/statistik/daten/studie/1073385/umfrage/anzahl-der-nutzer-von-messenger-apps-weltweit/ (besucht am 13. 07. 2023).

[2] A. Vasilaras, D. Dosis, M. Kotsis und P. Rizomiliotis, „Retrieving deleted records from telegram“, *Forensic Science International: Digital Investigation*, Jg. 43, Sep. 2022, ISSN: 26662817. DOI: 10.1016/j.fsidi.2022.301447.

[3] Bundeskriminalamt. „Cybergrooming“. (2023), Adresse: https://www.bka.de/DE/UnsereAufgaben/Aufgabenbereiche/Zentralstellen/Kinderpornografie/Cybergrooming/Cybergrooming_node.html (besucht am 13. 07. 2023).

[4] LF Projects. „About CASE“, Getting Started - just the basics. (2023), Adresse: https://caseontology.org/ontology/start.html (besucht am 28. 09. 2023).

[5] LF Projects. „Community Members“, A growing community working together to develop and implement CASE. (2023), Adresse: https://caseontology.org/community/members.html (besucht am 28. 09. 2023).

[6] M. Spranger, J. Xi, L. Jaeckel, J. Felser und D. Labudde, „MoNA: A forensic analysis platform for mobile communication“, *KI - Künstliche Intelligenz*, Jg. 36, Nr. 2, S. 163–169, Sep. 2022, ISSN: 0933-1875. DOI: 10.1007/s13218-022-00762-w.

[7] M. Moreb, *Practical Forensic Analysis of Artifacts on iOS and Android Devices: Investigating Complex Mobile Devices*. Berkeley, CA: Apress, 2022, ISBN: 978-1-4842-8025-6. DOI: 10.1007/978-1-4842-8026-3.

[8] R. Tamma und D. Tindall, *Learning Android forensics: a hands-on guide to Android forensics, from setting up the forensic workstation to analyzing key forensic artifacts* (Community experience distilled). Birmingham: Packt Publ, 2015, 291 S., ISBN: 978-1-78217-457-8.

[9] M. Epifani und P. Stirparo, *Learning iOS forensics: a practical hands-on guide to acquire and analyze iOS devices with the latest forensic techniques and tools* (Community experience distilled), 2. Aufl. Birmingham: Packt, 2016, 315 S., ISBN: 978-1-78588-208-1.

[10] S. Dogan und E. Akbal, „Analysis of mobile phones in digital forensics“, in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia: IEEE, Mai 2017, S. 1241–1244, ISBN: 978-953-233-090-8. DOI: 10.23919/MIPRO.2017.7973613.

[11] Bundesamt für Sicherheit in der Informationstechnik, „Leitfaden „IT-Forensik“ Version 1.0.1“, 15. März 2011. Adresse: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Cyber-Sicherheit/Themen/Leitfaden_IT-Forensik.pdf (besucht am 28. 09. 2023).

[12] R. L. Ackoff, „From data to wisdom“, *Journal of Applied Systems Analysis*, Jg. 16, S. 3–9, 1989.

[13] J. H. Bernstein, „The Data-Information-Knowledge-Wisdom Hierarchy and its Antithesis“, *NASKO*, Jg. 2, Nr. 1, S. 68, 4. Nov. 2011, ISSN: 2311-4487. DOI: 10.7152/nasko.v2i1.12806.

[14] R. J. Brachman, H. J. Levesque und M. Pagnucco, *Knowledge representation and reasoning*. Amsterdam: Morgan Kaufmann, 2004, OCLC: 147994324, ISBN: 978-0-08-048932-2.

[15] E. Sapir, *Language: An Introduction to the Study of Speech*. New York: Harcourt, Brace & World Inc., 1921.

[16] C. Ogden und I. Richards, *The Meaning of Meaning*. Oxford, England: Harcourt, Brace, 1923.

[17] D. Gaševic, D. Djuric und V. Devedžic, „Ontologies“, in *Model Driven Engineering and Ontology Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 45–80, ISBN: 978-3-642-00281-6. DOI: 10.1007/978-3-642-00282-3_2.

[18] T. R. Gruber, „A translation approach to portable ontology specifications“, *Knowledge Acquisition*, Jg. 5, Nr. 2, S. 199–220, Juni 1993, ISSN: 10428143. DOI: 10.1006/knac.1993.1008.

[19] B. Chandrasekaran, J. Josephson und V. Benjamins, „What are ontologies, and why do we need them?“, *IEEE Intelligent Systems*, Jg. 14, Nr. 1, S. 20–26, Jan. 1999, ISSN: 1094-7167. DOI: 10.1109/5254.747902.

[20] N. F. Noy und D. L. McGuinness, „Ontology Development 101: A Guide to Creating Your First Ontology“, *Knowledge Systems Laboratory*, Jg. 32, Jan. 2001.

[21] LF Projects. „Instance data guidance“. (2023), Adresse: https://caseontology.org/resources/instance_data.html (besucht am 24. 09. 2023).

[22] F. Gandon und G. Schreiber. „RDF 1.1 XML syntax“, W3C Recommendation. (25. Feb. 2014), Adresse: https://www.w3.org/TR/rdf-syntax-grammar/ (besucht am 24. 09. 2023).

[23] M. Dürst und M. Suignard. „Internationalized resource identifiers (IRIs)“. (Jan. 2005), Adresse: https://www.ietf.org/rfc/rfc3987.txt (besucht am 24. 09. 2023).

[24] E. Prud'hommeaux und G. Carothers. „RDF 1.1 turtle: Terse RDF triple language“, W3C Recommendation. (25. Feb. 2014), Adresse: https://www.w3.org/TR/turtle/ (besucht am 24. 09. 2023).

[25] D. Brickley und R. Guha. „RDF schema 1.1“, W3C Proposed Edited Recommendation. (9. Jan. 2014), Adresse: https://www.w3.org/TR/2014/PER-rdf-schema-20140109/ (besucht am 24. 09. 2023).

[26] W3C OWL Working Group. „OWL 2 web ontology language document overview (second edition)“, W3C Recommendation. (11. Dez. 2012), Adresse: https://www.w3.org/TR/owl2-overview/ (besucht am 24. 09. 2023).

[27] D. L. McGuinness und F. van Harmelen. „OWL web ontology language overview“, W3C Recommendation. (10. Feb. 2004), Adresse: https://www.w3.org/TR/owl-features/ (besucht am 24. 09. 2023).

[28] S. Harris und A. Seaborne. „SPARQL 1.1 query language“, W3C Recommendation. (21. März 2013), Adresse: https://www.w3.org/TR/sparql11-query/ (besucht am 24. 09. 2023).

[29] LF Projects. „About UCO“, Getting Started - just the basics. (2023), Adresse: https://unifiedcyberontology.org/ontology/start.html (besucht am 28. 09. 2023).

[30] LF Projects. „Subdomain Communities“. (2023), Adresse: https://unifiedcyberontology.org/adopters/subdomains.html (besucht am 28. 09. 2023).

[31] LF Projects. „Introduction“, What is CASE and where is it used? (2023), Adresse: https://caseontology.org/ontology/intro.html (besucht am 28. 09. 2023).

[32] LF Projects. „CASE Ontology Design and Specification“. (2023), Adresse: https://caseontology.org/resources/case_design_document.html (besucht am 28. 09. 2023).

[33] LF Projects. „FAQ“, Frequently Asked Questions. (2023), Adresse: https://caseontology.org/resources/faq.html (besucht am 28. 09. 2023).

[34] S. L. Garfinkel, „Digital forensics research: The next 10 years“, *Digital Investigation*, Jg. 7, S64–S73, Aug. 2010, ISSN: 17422876. DOI: 10.1016/j.diin.2010.05.009.

[35] A. Al-Dhaqm u. a., „Digital Forensics Subdomains: The State of the Art and Future Directions“, *IEEE Access*, Jg. 9, S. 152 476–152 502, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3124262.

[36] A. M. Hoss und D. L. Carver, „Weaving ontologies to support digital forensic analysis“, in *2009 IEEE International Conference on Intelligence and Security Informatics*, Richardson, TX, USA: IEEE, 2009, S. 203–205, ISBN: 978-1-4244-4172-3. DOI: 10.1109/ISI.2009.5137303.

[37] S. Garfinkel, „Digital forensics XML and the DFXML toolset“, *Digital Investigation*, Jg. 8, Nr. 3, S. 161–174, Feb. 2012, ISSN: 17422876. DOI: 10.1016/j.diin.2011.11.002.

[38] S. Lee, A. Savoldi, K. S. Lim, J. H. Park und S. Lee, „A proposal for automating investigations in live forensics“, *Computer Standards & Interfaces*, Jg. 32, Nr. 5, S. 246–255, Okt. 2010, ISSN: 09205489. DOI: 10.1016/j.csi.2009.09.001.

[39] B. N. Levine und M. Liberatore, „DEX: Digital evidence provenance supporting reproducibility and comparison“, *Digital Investigation*, Jg. 6, S. 48–56, Sep. 2009, ISSN: 17422876. DOI: 10.1016/j.diin.2009.06.011.

[40] W. Alink, R. Bhoedjang, P. Boncz und A. De Vries, „XIRAF – XML-based indexing and querying for digital forensics“, *Digital Investigation*, Jg. 3, S. 50–58, Sep. 2006, ISSN: 17422876. DOI: 10.1016/j.diin.2006.06.016.

[41] M. Cohen und B. Schatz, „Hash based disk imaging using AFF4“, *Digital Investigation*, Jg. 7, S. 121–128, Aug. 2010, ISSN: 17422876. DOI: 10.1016/j.diin.2010.05.015.

[42] G. Giova, „Improving Chain of Custody in Forensic Investigation of Electronic Digital Systems“, *Journal of Computer Science*, Jg. 11, Nr. 1, Jan. 2011.

[43] M. Morgenstern, J. Fähndrich und W. Honekamp, „Ontology in the digital forensics domain: A scoping review“, 2022, ISBN: 9783885797203 Herausgeber: Gesellschaft für Informatik, Bonn, ISSN: 1617-5468. DOI: 10.18420/INF2022_05.

[44] S. Dosis, I. Homem und O. Popov, „Semantic representation and integration of digital evidence“, *Procedia Computer Science*, Jg. 22, S. 1266–1275, 2013, ISSN: 18770509. DOI: 10.1016/j.procs.2013.09.214.

[45] B. Schatz, G. Mohay und A. Clark, „Rich Event Representation for Computer Forensics“, *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference 2004*, S. 1–16, Juni 2004.

[46] B. Schatz, G. Mohay und A. Clark, „Generalising Event Correlation Across Multiple Domains“, *Journal of Information Warfare*, Jg. 4, Nr. 1, S. 69–79, 2005, Herausgeber: Peregrine Technical Solutions, ISSN: 14453312.

[47] D. Kahvedžić und T. Kechadi, „DIALOG: A framework for modeling, analysis and reuse of digital forensic knowledge“, *Digital Investigation*, Jg. 6, S. 23–33, Sep. 2009, ISSN: 17422876. DOI: 10.1016/j.diin.2009.06.014.

[48] D. Ellison, R. A. Ikuesan und H. S. Venter, „Ontology for Reactive Techniques in Digital Forensics“, in *2019 IEEE Conference on Application, Information and Network Security (AINS)*, Pulau Pinang, Malaysia: IEEE, Nov. 2019, S. 83–88, ISBN: 978-1-72813-306-5. DOI: 10.1109/AINS47559.2019.8968696.

[49] ISO/IEC JTC 1/SC 27, „ISO/IEC 27043:2015 - Information technology — Security techniques — Incident investigation principles and processes“, ISO und IEC, März 2015. Adresse: https://www.iso.org/standard/44407.html (besucht am 13. 10. 2023).

[50] D. C. Harrill und R. P. Mislan, „A Small Scale Digital Device Forensics ontology“, *Small Scale Digital Device Forensics Journal*, Jg. 1, Nr. 1, Juni 2007.

[51] B. Turnbull und S. Randhawa, „Automated event and social network extraction from digital evidence sources with ontological mapping“, *Digital Investigation*, Jg. 13, S. 94–106, Juni 2015, ISSN: 17422876. DOI: 10.1016/j.diin.2015.04.004.

[52] A. Akremi, M.-F. Sriti, H. Sallay und M. Rouached, „Ontology-Based Smart Sound Digital Forensics Analysis for Web Services:“ In *Digital Forensics and Forensic Investigations*, I. R. Management Association, Hrsg., IGI Global, 2020, S. 497–520, ISBN: 978-1-79983-025-2. DOI: 10.4018/978-1-7998-3025-2.ch033.

[53] H. Arshad, A. Jantan, G. K. Hoon und I. O. Abiodun, „Formal knowledge model for online social network forensics“, *Computers & Security*, Jg. 89, Feb. 2020, ISSN: 01674048. DOI: 10.1016/j.cose.2019.101675.

[54] M. Alzaabi, T. Martin, K. Taha und A. Jones, „The Use of Ontologies in Forensic Analysis of Smartphone Content“, *Journal of Digital Forensics, Security and Law*, 2015, ISSN: 15587223. DOI: 10.15394/jdfsl.2015.1215.

[55] E. Casey, G. Back und S. Barnum, „Leveraging CybOX™ to standardize representation and exchange of digital forensic information“, *Digital Investigation*, Jg. 12, S. 102–110, März 2015, ISSN: 17422876. DOI: 10.1016/j.diin.2015.01.014.

[56] E. Casey, S. Barnum, R. Griffith, J. Snyder, H. Van Beek und A. Nelson, „Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language“, *Digital Investigation*, Jg. 22, S. 14–45, Sep. 2017, ISSN: 17422876. DOI: 10.1016/j.diin.2017.08.002.

[57] E. Casey, S. Barnum, R. Griffith, J. Snyder, H. Van Beek und A. Nelson, „The Evolution of Expressing and Exchanging Cyber-Investigation Information in a Standardized Form“, in *Handling and Exchanging Electronic Evidence Across Europe*, Ser. Law, Governance and Technology Series, M. A. Biasiotti, J. P. Mifsud Bonnici, J. Cannataci und F. Turchi, Hrsg., Bd. 39, Cham: Springer International Publishing, 2018, S. 43–58, ISBN: 978-3-319-74871-9. DOI: 10.1007/978-3-319-74872-6_4.

[58] E. Casey, S. Barnum, R. Griffith, J. Snyder, H. Van Beek und A. Nelson, „Corrigendum to ‘advancing coordinated cyber-investigations and tool interoperability using a community developed specification language’ [digital investigation 22c (2017) 14–45]“, *Digital Investigation*, Jg. 28, S. 183–187, März 2019. DOI: 10.1016/j.diin.2018.10.001.

[59] E. Casey, A. Nelson und J. Hyde, „Standardization of file recovery classification and authentication“, *Digital Investigation*, Jg. 31, Dez. 2019, ISSN: 17422876. DOI: 10.1016/j.diin.2019.06.004.

[60] Adoption Committee Mapping Working Group, „Cyber-Investigation Analysis Standard Expression (CASE) Mapping Guide“, Aug. 2022. Adresse: https://drive.google.com/file/d/11bxU-BUHd--JPUsRsL5jTI1fBcsgNy2E/view (besucht am 28. 09. 2023).

[61] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, P.-A. Champin und N. Lindström. „JSON-LD 1.1 - a JSON-based serialization for linked data“, W3C Recommendation. (16. Juli 2020), Adresse: https://www.w3.org/TR/json-ld11/ (besucht am 24. 09. 2023).

[62] R. Rivest, „The MD5 message-digest algorithm“, RFC1321, Apr. 1992. DOI: 10.17487/rfc1321.

[63] D. Steinberg, F. Budinsky, M. Paternostro und E. Merks, *EMF: Eclipse Modeling Framework*, 2. Aufl. Addison Wesley, 2009, OCLC: 1156845233, ISBN: 978-0-13-270221-8.

[64] M. A. Musen, „The protégé project: A look back and a look forward“, *AI Matters*, Jg. 1, Nr. 4, S. 4–12, 16. Juni 2015, ISSN: 2372-3483. DOI: 10.1145/2757001.2757003.

[65] Digital Ocean, Nutzer e09330e1749e4191a9dfe441e5129f. „A Complete Guide to Install Tomcat on Linux“, Tutorial. (3. Aug. 2022), Adresse: https://www.digitalocean.com/community/tutorials/install-tomcat-on-linux (besucht am 09. 10. 2023).

[66] The Apache Software Foundation. „Apache Jena Fuseki“, Documentation. (2023), Adresse: https://jena.apache.org/documentation/fuseki2/index.html (besucht am 09. 10. 2023).

[67] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur und Y. Katz, „Pellet: A practical OWL-DL reasoner“, *Journal of Web Semantics*, Jg. 5, Nr. 2, S. 51–53, Juni 2007, ISSN: 15708268. DOI: 10.1016/j.websem.2007.03.004.

[68] B. Glimm, I. Horrocks, B. Motik, G. Stoilos und Z. Wang, „HermiT: An OWL 2 reasoner“, *Journal of Automated Reasoning*, Jg. 53, Nr. 3, S. 245–269, Okt. 2014, ISSN: 0168-7433. DOI: 10.1007/s10817-014-9305-1.

Eidesstattliche Erklärung

Hiermit versichere ich – Lukas Jaeckel – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Mittweida, 20. Oktober 2023	
Ort, Datum	Lukas Jaeckel, B.Sc.