Ganen Sethupathy
Stephan Zelewski
Jan Schagen

Wissensbasiertes IT-Projektmanagement mit KI- und Cloud-Techniken

Systematische Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte auf der Basis von Ontologien, Case-based Reasoning und Word2Vec

λογος

Ganen Sethupathy, Stephan Zelewski, Jan Peter Schagen

Wissensbasiertes IT-Projektmanagement mit KI- und Cloud-Techniken

Systematische Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte auf der Basis von Ontologien, Case-based Reasoning und Word2Vec

Logos Verlag Berlin

λογος

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Umschlagabbildung: iStock, Peach_iStock

Logos Verlag Berlin GmbH 2024

ISBN 978-3-8325-5883-3

Logos Verlag Berlin GmbH
Georg-Knorr-Str. 4, Geb. 10,
12681 Berlin, Germany

Tel.: +49 (0)30 / 42 85 10 90
Fax: +49 (0)30 / 42 85 10 92
http://www.logos-verlag.de

Inhaltsverzeichnis

1 Einführung in die Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

1.1 Realprobleme der Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

Die hier vorgelegten Untersuchungen[1] befassen sich mit der hohen Bedeutung von sicherheitskritischen IT-Systemen für Unternehmen aus der gewerblichen Wirtschaft, öffentliche Institutionen und die zivile Bevölkerung. Diese Bedeutung ist in den letzten Jahren stark angestiegen, wie in Kürze anhand zweier prägnanter Beispiele erläutert wird.

Sicherheitskritisch ist ein IT-System, das für die Sicherheit der zivilen Bevölkerung als relevant angesehen und dessen möglicher Ausfall mit einer Gefährdung der öffentlichen Sicherheit verbunden wird. Unter sicherheitskritische IT-Systeme fallen auf Informationstechnik basierende kritische Infrastrukturen, wie beispielsweise ein Einsatzleitsystem oder der Digitalfunk der Behörden und Organisationen mit Sicherheitsaufgaben. Sicherheitskritische IT-Systeme fallen in den Bereich der Kritischen Infrastrukturen (KRITIS). Sie sind definiert als „Organisationen oder Einrichtungen mit wichtiger Bedeutung für das staatliche Gemeinwesen, bei deren Ausfall oder Beeinträchtigung nachhaltig wirkende Versorgungsengpässe, erhebliche Störungen der öffentlichen Sicherheit oder andere dramatische Folgen eintreten würden“ (BUNDESMINISTERIUM DES INNERN (2009), S. 3).

An sicherheitskritische IT-Systeme werden neben ökonomischen auch gesellschaftliche Anforderungen gestellt. Der Ausfall oder die verspätete Inbetriebnahme eines sicherheitskritischen IT-Systems kann Menschenleben gefährden, schwerwiegende wirtschaftliche Folgen nach sich ziehen sowie das Sicherheitsbedürfnis der zivilen Bevölkerung nachhaltig beeinflussen. Die Bereitstellung sicherheitskritischer IT-Systeme erfolgt durch sicherheitskritische IT-Projekte. Für die erfolgreiche Durchführung sicherheitskritischer IT-Projekte gilt die Wiederverwendung von bestehendem Wissen, sogenanntem Erfahrungswissen, als ein kritischer Erfolgsfaktor. Probleme, die durch fehlendes Erfahrungswissen bei sicherheitskritischen IT-Projekten entstehen, können zum Scheitern von sicherheitskritischen IT-Projekten oder zu mangelnder Wertschöpfung eines Projektes führen. Die nachstehenden zwei Praxisbeispiele aus Europa verdeutlichen dies.

---

1) Dieser Beitrag beruht im Wesentlichen auf der Dissertation des erstgenannten Verfassers; vgl. SETHUPATHY (2024). Überarbeitungen – insbesondere eine deutliche inhaltliche Straffung sowie die englischsprachige Übersetzung – erfolgten durch den zweit- und den drittgenannten Verfasser (sowie in deren Auftrag durch eine professionelle Übersetzungsagentur). Der zugrunde liegenden (deutschsprachigen) Publikation SETHUPATHY (2024) können vor allem umfangreiche ergänzende Details, ein sehr ausführlicher Anhang vor allem mit ausführlichen Angaben zur Ontologie und zum CBR-System sowie eine Fülle von vertiefenden Quellenangaben entnommen werden. Insbesondere mit Quellenangaben in Fußnoten wurde, wie in „modernen“, internationalen Publikationen üblich, in diesem Beitrag bewusst sehr sparsam umgegangen, um den Lesefluss und das optische Erscheinungsbild des Layouts nicht allzu stark zu unterbrechen. Leser, die an wesentlich ausführlicheren Quellenangaben einschließlich zugehöriger Quellenkritik interessiert sind, werden gebeten, die Dissertation des erstgenannten Verfassers – vgl. SETHUPATHY (2024) – direkt zu konsultieren.

Das sicherheitskritische IT-Projekt „Kooperative Leitstelle Berlin“, welches sich gegenwärtig in der Implementierungsphase befindet, hat aufgrund von Planungsfehlern und einer Vergabeverzögerung gestiegene Kosten von ursprünglich 84 Millionen Euro auf gegenwärtig rund 250 Millionen Euro. Als zusätzliche Folge verzögert sich das Projekt um voraussichtlich sieben Jahre. Da die Wartung der jetzigen Leitstellen nur noch bis 2025 sichergestellt werden kann, wird eine alternative Lösung für die Überbrückungszeit notwendig, was weitere Kosten verursacht und auch das Sicherheitsgefühl der Gesellschaft hinsichtlich der Notrufbewältigung gefährdet.

Ein weiteres Beispiel ist die Erneuerung des Schweizer Sicherheitsfunknetzes Polycom, bei dem es durch unterschätzte Anforderungen und Mängel zu einer Projektverzögerung kam. Dies führt aufgrund der kritischen Zeitplanung und eines damit verbundenen möglicherweise notwendigen Parallelbetriebs des neuen und alten Systems zu einem höheren Risiko für Mehrkosten. Das Projekt erfährt aufgrund dessen und der direkten Vergabe ohne öffentliche Ausschreibung eine öffentliche Diskussion mit der Gefahr eines wachsenden Misstrauens in der Gesellschaft. Der Auftraggeber gab öffentlich bekannt, dass es an fundiertem und spezifischem Fachwissen fehlte. In einer Pressemitteilung hierzu heißt es: „Dieses Wissen aufzubauen hat sich aufseiten des Auftragnehmers als komplexer und schwieriger als ursprünglich angenommen herausgestellt.“ (DER BUNDESRAT DER SCHWEIZER REGIERUNG (2021).

Die Praxisbeispiele verdeutlichen, dass der Einsatz von Fach- und Erfahrungswissen einen kritischen Erfolgsfaktor für sicherheitskritische IT-Projekte darstellt. In diesem Zusammenhang kann auch von einem „project knowledge management“ gesprochen werden, weil das Projektmanagement eine besonders wissensintensive Managementaufgabe darstellt. Die enge Verbindung zwischen Wissens- und Projektmanagement wird auch in verschiedenen Publikationen und durch gängige Projektmanagementstandards (wie z. B. PRINCE2, PMI, Scrum und IPMA) verdeutlicht. Die Standards empfehlen, aus bereits abgeschlossenen Projekten Erfahrungen zu ziehen, die zum Beispiel im Rahmen von Lessons Learned, Sprint Retrospectives und Phasenabschlussberichten dokumentiert werden. Dieses Erfahrungswissen sollte in ähnlichen Projekten oder in weiteren Sprints eingesetzt werden, um an positiven und negativen Erfahrungen partizipieren zu können.

Die Relevanz der Wiederverwendung von Erfahrungswissen einerseits und dessen unzureichende Berücksichtigung im betrieblichen Alltag andererseits verdeutlichen das problemorientierte Spannungsfeld der Wiederverwendung von Erfahrungswissen im Projektmanagement im Allgemeinen und im Projektmanagement sicherheitskritischer IT-Projekte im Besonderen. Dieses *Wiederverwenungsproblem* umfasst vier Subprobleme, die nachfolgend kurz aufgelistet und alsdann hinsichtlich ihrer inhaltlichen Ausgestaltung und weiterer subsumierter Subsubprobleme erläutert werden.

Zunächst wird das *Wissensrepräsentationsproblem* betrachtet. Die Speicherung von Erfahrungswissen im betrieblichen Projektmanagement erfolgt meist in Form von Dokumenten, Plänen oder Protokollen. Diese Form der Speicherung ist jedoch aus verschiedenen Gründen problematisch.

- Strukturierungsproblem: Die Dokumente liegen zumeist in unstrukturierter Form vor.
- Quellenproblem: In einem betrieblichen Umfeld existieren darüber hinaus unterschiedliche Quellen, die für die Ablage solcher Dokumente in Frage kommen, weswegen kein zentraler Ablageort für die Speicherung von Erfahrungswissen existiert.
- Vokabularproblem: Die Dokumentation des Erfahrungswissens von unterschiedlichen Akteuren erfolgt heterogen, sodass die Dokumente über kein einheitlich definiertes Vokabular verfügen, welches von allen Akteuren auf gleiche Weise verstanden wird.

Das *Systemproblem* zeigt sich im Fehlen eines geeigneten IT-Systems für die Wiederverwendung von Erfahrungswissen aus sicherheitskritischen IT-Projekten. Die mangelnde Eignung lässt sich in problemorientierter Diktion wie folgt unterteilen:

- Verarbeitungsproblem: Bestehende IT-Systeme können unstrukturiertes Wissen schwer verarbeiten.
- Zugänglichkeitsproblem: Für die Akzeptanz eines IT-Systems ist es darüber hinaus notwendig, dass jederzeit und an jedem Ort auf Erfahrungswissen zugegriffen werden kann. Dies ist gegenwärtig nicht möglich.
- Intelligenzproblem: In der Regel beschränken sich vorliegende IT-Systeme auf die reine Anzeige der Dokumente. Es findet keine „intelligente“[2] Verarbeitung der Dokumentinhalte statt.

Das *Wissensverlustproblem* umfasst den Verlust von Erfahrungswissen. Der Verlust von Erfahrungswissen kann aus unterschiedlichen Gründen erfolgen, die in problemorientierter Diktion wie folgt unter dem Wissensverlustproblem subsumiert werden:

- Projektteamwechselproblem: Ständig neu zusammengestellte Projektteams erzeugen zwar Dynamik, führen aber zum Verlust des Erfahrungswissens von Spezialisten.
- Kündigungsproblem: Geplante oder ungeplante Unternehmensaustritte können zu einem Verlust von Wissen führen.

Das *Wissensauswertungsproblem* umfasst alle Probleme, die mit der Auswertung von bestehendem Erfahrungswissen zusammenhängen:

- Ressourcenproblem: Die Suche nach Erfahrungswissen dauert in den meisten IT-Systemen zu lange und ist zu ressourcenintensiv, da heterogene Quellen und natürlichsprachliche Texte durchsucht werden müssen.
- Technikproblem: Mit bestehenden Techniken können natürlichsprachliche Wissenskomponenten nur unzureichend identifiziert und automatisiert ausgewertet werden.

2) Als „intelligent“ wird in diesem Beitrag ein IT-System verstanden, das über die rein *syntaktische* Suche (z. B. durch String-Matching) hinausgeht. „Intelligent“ ist ein IT-System demnach, wenn mit seiner Hilfe inhaltsadressiert, also mit „inhaltlichem Verständnis“ und somit *semantisch* nach wiederverwendbarem Erfahrungswissen gesucht werden kann.

- Vergleichsproblem: Die Vergleichbarkeit von sicherheitskritischen IT-Projekten gestaltet sich problematisch, da kein Maßstab zum systematischen Vergleich sicherheitskritischer IT-Projekte existiert.

1.2 Betriebswirtschaftliche Desiderate zur Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

Auf Basis der vorherigen Ausführungen lassen sich die betriebswirtschaftlichen Desiderate ableiten. Das übergeordnete Desiderat, nachfolgend *Wiederverwendungsdesiderat* genannt, lautet, dass es wünschenswert wäre, eine Möglichkeit zur intelligenten Wiederverwendung von Erfahrungswissen von sicherheitskritischen IT-Projekten zu schaffen. Aus diesem lassen sich die folgenden Teildesiderate ableiten:

- Wissensrepräsentationsdesiderat
- Systemdesiderat
- Wissensauswertungsdesiderat
- Wissensverlustdesiderat

Das *Wissensrepräsentationsdesiderat* drückt den Bedarf an einer vollumfänglichen Repräsentation von Erfahrungswissen aus und umfasst nachfolgende Subdesiderate:

- Strukturierungsdesiderat: Es wäre wünschenswert, Erfahrungswissen zu strukturieren.
- Quellendesiderat: Es wäre wünschenswert, Erfahrungswissen zentral an einer Quelle zu speichern.
- Vokabulardesiderat: Es wäre wünschenswert, Erfahrungswissen mittels eines einheitlichen Vokabulars zu repräsentieren.

Das *Systemdesiderat* drückt aus, dass es erstrebenswert wäre, ein IT-System bereitzustellen, welches die Wiederverwendung von Erfahrungswissen ermöglicht. Das Systemdesiderat umfasst nachfolgende Subdesiderate:

- Verarbeitungsdesiderat: Es wäre wünschenswert, unstrukturiertes Wissen verarbeiten zu können.
- Zugänglichkeitsdesiderat: Es wäre wünschenswert, Erfahrungswissen überall und jederzeit zugänglich machen zu können.
- Intelligenzdesiderat: Es wäre wünschenswert, Erfahrungswissen „intelligent“ wiederzuverwenden.

Das *Wissensverlustdesiderat* drückt aus, dass es erstrebenswert wäre, jegliches Erfahrungswissen zu speichern, um einen Wissensverlust zu verhindern. Das Wissensverlustdesiderat umfasst nachfolgenden Subdesiderate:

- Projektteamwechseldesiderat: Es wäre wünschenswert, Erfahrungswissen von ausscheidenden Projektmitarbeitern bei Projektteamwechseln speichern zu können.

- Kündigungsdesiderat: Es wäre wünschenswert, Erfahrungswissen von ausscheidenden Mitarbeitern speichern zu können.

Das *Wissensauswertungsdesiderat* drückt den Bedarf an einer systematischen Auswertung von gespeichertem Erfahrungswissen aus und umfasst nachfolgende Subdesiderate:

- Ressourcendesiderat: Es wäre wünschenswert, wenn die Auswertung von Erfahrungswissen effektiv und effizient vollzogen werden kann.
- Technikdesiderat: Es wäre wünschenswert, Techniken zu inkludieren, die auch natürlichsprachliche Wissenskomponenten auswerten können.
- Vergleichsdesiderat: Es wäre wünschenswert, wenn systematische Vergleichsmaßstäbe für die Auswertung von sicherheitskritischen IT-Projekten vorliegen.

1.3 State of the Art der verfügbaren Techniken zur Erfüllung der betriebswirtschaftlichen Desiderate

Die Betrachtung des State of the Art zeigt, dass die Kombination von Ontologien und Case-based Reasoning (CBR) grundsätzlich zur intelligenten Wiederverwendung von Erfahrungswissen im Projektmanagement geeignet ist. Beim ontologiegestützten Case-based Reasoning werden die zwei Techniken Ontologien und Case-based Reasoning kombiniert. Der Einsatz von Ontologien in Case-based-Reasoning-Systemen (kurz: CBR-Systemen) ist nicht neu. Er wurde bereits in verschiedenen Forschungspublikationen behandelt; vgl. DUARTE/BELO (2023), S. 830-842; NKISI-ORJI et al. (2022), S 127-138; OBEID et al. (2022), S. 991-1002; WANG/LIN/ZHANG (2022), S. 4-19; WEBER et al. (2021), S. 12-27; EMMENEGGER et al. (2017), S. 338-351; MARTIN et al. (2017), S. 552-571; BOUHANA et al. (2015), S. 3726-3740; ZELEWSKI/KOWALSKI/BERGENRODT (2015a), S. 294-302; ZELEWSKI/KOWALSKI/BERGENRODT (2015b), S. 242-255; RECIO-GARCÍA/GONZÁLEZ-CALERO/DÍAZ-AGUDO (2014), S. 129, 134 u. 137-143; AMAILEF/LU (2013), S. 81-96; GUO/HU/PENG (2012), S. 497-507; BEIßEL (2011), S. 40-220; ASSALI/LENNE/DEBRAY (2010), S. 97-115; WYNER (2008), S. 361-385; RECIO-GARCÍA et al. (2007), S. 151-161; BERGMANN/SCHAAF (2003), S. 608-624. Dieser Einsatz von Ontologien in CBR-Systemen wird grundsätzlich als positiv erachtet; vgl. WANG/LIN/ZHANG (2022), S. 5; BEIßEL (2011), S. 40. Der Vorteil der Kombination von Ontologien mit CBR-Systemen wird vor allem darin gesehen, dass eine Ontologie das einheitliche Vokabular vorgibt, das in einem CBR-System eingesetzt wird. Präziser sind die sprachlichen Ausdrucksmittel gemeint, die in einem ontologiegestützten CBR-System zur Problemspezifizierung und Problemlösung eingesetzt werden können. Diese sprachlichen Ausdrucksmittel können über ein simples Vokabular hinausgehen, wie z. B. nicht-taxonomische Relationen sowie Inferenz- und Integritätsregeln; darauf wird später ausführlicher zurückgekommen.

Zunächst werden die beiden Techniken getrennt voneinander beschrieben und danach wird die Kombination des ontologiegestützten Case-based Reasonings erläutert.

Ontologien ermöglichen es, ein gemeinsam geteiltes Verständnis über die verwendeten sprachlichen Ausdrucksmittel, insbesondere „Begrifflichkeiten“ (neben z. B. zwischenbegrifflichen semantischen Relationen und semantischen Axiomen hinsichtlich der inhaltlich korrekten Begriffsverwendung) zu schaffen. Diese sprachlichen Ausdrucksmittel werden in Ontologien konzeptualisiert und formal definiert; vgl. CAROLLA (2015), S. 31.

Ontologien, die sich auf die Domäne des Projektmanagements beziehen, sind im State of the Art bereits zu finden; vgl. SANTOS JÚNIOR et al. (2021), S. 7-25, mit einem Fokus auf agile Projekte; WEBER et al. (2021), S. 12-23 u. 50-75, mit einem Fokus auf sicherheitskritische IT-Projekte; MARTIN et al. (2017), S. 551-552, 562 u. 567-570; ZELEWSKI/KOWALSKI/BERGENRODT (2015b), S. 245-250; LIN et al. (2012), S. 195-206; SHEEBA/KRISHNAN/BERNARD (2012), S. 2-7; DONG/HUSSAIN/CHANG (2011), S. 1164-1169; HUGHES (2010), S. 9-19; ARAMO-IMMONEN (2009), S. 49-55; SARANTIS/ASKOUNIS (2009), S. 2-7; ABELS et al. (2006), S. 817-819. Die Domäne sicherheitskritischer IT-Projekte wird gegenwärtig jedoch noch nicht gesondert fokussiert. Lediglich WEBER et al. (2021) haben einen ersten Versuch unternommen, eine Projektmanagementontologie für sicherheitskritische IT-Projekte zu konzeptualisieren.

Das Case-based Reasoning unterliegt dem Grundgedanken, neue Fälle mit den bereits gelösten Fällen abzugleichen und den geeignetsten Fall auf den neuen Fall anzuwenden. Im Bereich des Projektmanagements bedeutet dies, dass die Wiederverwendung des Erfahrungswissens aus vorherigen, möglichst ähnlichen Projekten für die Durchführung neuer Projekte ermöglicht werden soll. Jedoch beschränken sich die aktuellen Einsatzfelder des Case-based Reasonings als isolierte Methode für das Projektmanagement auf quantitative Wissenskomponenten, wie beispielweise für die Abschätzung von Projektkosten; vgl. beispielsweise RADZIEJOWSKA/ZIMA (2015), S. 100-111; ZIMA (2015), S. 59-64; KIM/SHIM (2014), S. 66-72; JI/PARK/LEE (2012), S. 45-51; KIM et al. (2012), S. 284-291. Projektspezifisches, insbesondere qualitatives (natürlichsprachliches) Erfahrungswissen, wie etwa Lasten- und Pflichtenhefte oder Mitarbeiterkompetenzen, werden dagegen nur selten oder nicht ausreichend berücksichtigt.

Die Kombination der beiden KI-Techniken „Ontologien“ und „Case-based Reasoning“, das ontologiegestützte Case-based Reasoning, kann für das Wissensmanagement im Projektmanagement Erfolg versprechend eingesetzt werden. Für seine Umsetzung existieren bereits einige CBR-Tools zur Konstruktion ontologiegestützter CBR-Systeme. Beispielhaft zu erwähnen sind die CBR-Tools jCOLIBRI, MyCBR, COBRA, CASBIAN, CBR-Shell, Induce-IT und KAIDARA Advisor; vgl. beispielsweise MARTIN et al. (2017), S. 556-557.

Ein CBR-Tool, das speziell für die Anwendung des ontologiegestützten Case-based Reasonings im Projektmanagement entwickelt wurde, ist „jCORA“ (java based Case- and Ontology-based Reasoning Application). WEBER et al. (2021), S. 23-43, behandeln den Einsatz des CBR-Tools jCORA speziell für sicherheitskritische IT-Projekte. Jedoch reicht die Projektontologie an Ausdrucksstärke nicht aus, um sicherheitskritische IT-Projekte aus betrieblicher Sicht ausreichend darzustellen. Insbesondere wurden in der Publikation von WEBER et al. keine Leistungsbe-

schreibungen und Bekanntmachungen aus sicherheitskritischen IT-Projekten für die Konzeptualisierung zugrunde gelegt. Des Weiteren sind die Begrifflichkeiten aus dem PRINCE2-Standard in der Ontologie nicht gemäß dem PRINCE2-Standard definiert.

Ein aktuelles Forschungsprojekt zu einem CBR-System, das auf Cloud-Technologien basiert, ist das Projekt CLOOD; vgl. NKISI-ORJI et al. (2022), S. 125-138; NKISI-ORJI et al. (2020), S. 132-142. Das Forschungsprojekt stellt die Phasen des CBR-Zyklus als eigenständige, öffentlich einsehbare und nutzbare Serverless-Funktionen in der Programmiersprache Python bereit, sodass die Funktionen für jegliche Cloud-basierten CBR-Implementierungsprojekte genutzt werden können. Eine Umsetzung für die Domäne Projektmanagement oder sicherheitskritische IT-Projekte existiert allerdings zurzeit für ein Cloud-basiertes, ontologiegestütztes Case-based Reasoning noch nicht.

Ähnlichkeitsmaßstäbe für die Verarbeitung von qualitativen Wissenskomponenten erfolgen in einem ontologiegestützten CBR-System mittels Ähnlichkeitsalgorithmen, die zur Ermittlung von Ähnlichkeiten insbesondere auf die semantische Distanz innerhalb von Taxonomiegraphen einer Ontologie zurückgreifen. Alternativ finden Ähnlichkeitstabellen häufig Anwendung, jedoch unterliegt deren konkrete Ausgestaltung eher recht subjektiven Heuristiken ohne methodischen Hintergrund.

Künstliche Neuronale Netze (KNN) bieten Ansatzpunkte für die Ähnlichkeitsermittlung von Begriffen an. Zwar bietet die bereits erwähnte Publikation von NKISI-ORJI eine erste Grundlage für die Verbindung zwischen ontologiegestützten CBR-Systemen und Künstlichen Neuronalen Netzen an, jedoch bleibt die Publikation vage hinsichtlich eines Implementierungsvorschlags sowie für die Aufbereitung der Trainingsdaten.

1.4 Wissenschaftliche Probleme im Hinblick auf die Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

Nachfolgend wird jedem betriebswirtschaftlichen Desiderat auf der untersten Ebene der Desideratehierarchie der jeweilige State of the Art gegenübergestellt, um zu überprüfen, ob jeweils eine nicht-triviale Diskrepanz vorliegt.

Zunächst wird die nicht-triviale Diskrepanz zwischen den Subdesideraten des Systemdesiderats und dem gegenübergestellten State of the Art für ontologiegestütztes Case-based Reasoning beschrieben.

- *Strukturierungsdesiderat:* Durch das Verwenden einer Ontologie in einem CBR-System ist es möglich, Erfahrungswissen strukturiert zu speichern. Es liegt jedoch keine Ontologie vor, welche hierfür ausreichendes Vokabular für sicherheitskritische IT-Projekte zur Verfügung stellt.

- *Quellendesiderat:* In einem CBR-System kann Erfahrungswissen aus sicherheitskritischen IT-Projekten zentral in einem CBR-System gespeichert werden. Es existiert jedoch bisher keine ausreichende Implementierung eines zentralen ontologiegestützten CBR-Systems für sicherheitskritische IT-Projekte.
- *Vokabulardesiderat:* Durch die Verwendung einer Ontologie kann Erfahrungswissen mittels eines zuvor definierten einheitlichen Vokabulars gespeichert werden. Es liegt jedoch keine Ontologie vor, die ein ausreichendes Vokabular für sicherheitskritische IT-Projekte zur Verfügung stellt.
- *Intelligenzdesiderat:* Grundsätzlich sind ontologiegestützte CBR-Systeme in der Lage, Erfahrungswissen intelligent wiederzuverwenden. Für die Domäne der sicherheitskritischen IT-Projekte liegt jedoch keine Ontologie vor, in der die notwendigen sprachlichen Ausdrucksmittel spezifiziert wurden, welche die intelligente Wiederverwendung von Erfahrungswissen in dieser Domäne ermöglicht. Die im State of the Art betrachteten CBR-Tools, insbesondere jCORA, können aufgrund der Einschränkungen hinsichtlich der eingesetzten Softwaretechnologie und der Usability in der betrieblichen Praxis kaum eingesetzt werden. Es zeigt sich, dass die CBR-Systeme von der betrieblichen Praxis als unzureichend empfunden werden.
- *Verarbeitungsdesiderat:* Grundsätzlich unterstützen ontologiegestützte CBR-Systeme die Verarbeitung von qualitativen Wissenskomponenten, jedoch erfolgt bisher keine vollumfassende Anwendung für sicherheitskritische IT-Projekte, da für die Domäne der sicherheitskritischen IT-Projekte keine Ontologie vorliegt, in der die notwendigen sprachlichen Ausdrucksmittel spezifiziert wurden.
- *Zugänglichkeitsdesiderat:* Grundsätzlich kann auf CBR-Tools für die Konstruktion ontologiegestützter CBR-Systeme, wie z. B. jCORA, ubiquitär zugegriffen werden. Jedoch unterliegt die ubiquitäre Nutzung einigen Anwendungsbarrieren, beispielsweise der notwendigen Installation einer Java-Umgebung sowie dem ausschließlichen Zugriff über einen Desktop-Client. Um eine ubiquitäre Nutzung zu ermöglichen, bestehen erste Ansätze, Cloud-basierte CBR-Systeme, im Einzelfall auch ontologiegestützte CBR-Systeme, zu entwickeln. Diese Ansätze lassen sich jedoch nicht als ausgereifte Systeme auffassen, die im Kontext von sicherheitskritischen IT-Projekten genutzt werden können.
- *Kündigungsdesiderat:* Grundsätzlich ermöglicht ein ontologiegestütztes CBR-System die Speicherung von Erfahrungswissen von aus einem Unternehmen ausscheidender Mitarbeiter. Für die Domäne der sicherheitskritischen IT-Projekte liegt keine Ontologie vor, in der die notwendigen sprachlichen Ausdrucksmittel modelliert wurden.
- *Projektteamwechseldesiderat:* Grundsätzlich ermöglicht ein ontologiegestütztes CBR-System die Speicherung von Erfahrungswissen der aus einem Projektteam ausscheidenden Mitarbeiter. Für die Domäne der sicherheitskritischen IT-Projekte liegt keine Ontologie vor, in der die notwendigen sprachlichen Ausdrucksmittel spezifiziert wurden.

- *Ressourcendesiderat:* Grundsätzlich ist eine Auswertung von Erfahrungswissen mit den CBR-Systemen, die mit üblichen CBR-Tools wie jCORA konstruiert wurden, möglich. Jedoch verursachen die Bearbeitungsschritte, wie beispielsweise die Ähnlichkeitsberechnung, eine hohe lokale Berechnungslast, die aufgrund der monolithischen Anwendungsstruktur unzureichend auf mehrere Rechenressourcen verteilt werden kann. Cloud-native-Anwendungen bieten hinsichtlich der Verteilbarkeit der Berechnungslast Lösungsmöglichkeiten an, die in den gängigen monolithischen CBR-Tools fehlen. Eine mögliche Implementierung des ontologiegestützten Case-based Reasonings als Cloud-Native-Anwendung für sicherheitskritische IT-Projekte existiert gegenwärtig nicht.
- *Technikdesiderat:* Die Auswertung von natürlichsprachlichen Wissenskomponenten ist durch ein ontologiegestütztes CBR-System zwar grundsätzlich möglich, jedoch ist insbesondere die Nutzung von Ähnlichkeitstabellen in der Praxis nicht immer effektiv und effizient. Künstliche Neuronale Netze, die durch Cloud-native-Anwendungen genutzt werden können, bieten bei der Auswertung von natürlichsprachlichen Wissenskomponenten Ansatzpunkte auf Basis von Vergleichsmaßstäben an.
- *Vergleichsdesiderat:* CBR-Systeme, die mit üblichen CBR-Tools wie jCORA konstruiert wurden, bieten universelle Vergleichsmaßstäbe für den Vergleich von Projekten an. Spezifische Vergleichsmaßstäbe für sicherheitskritische IT-Projekten existieren aktuell nicht. Aufgrund der monolithischen Anwendungsstruktur sind die Ähnlichkeitsfunktionen für die Vergleichsmaßstäbe zudem an die CBR-Systeme gebunden. Erweiterungen, wie die Implementierung von zusätzlichen spezifischen Ähnlichkeitsfunktionen für sicherheitskritische IT-Projekte, bedingen eine Entwicklung in den CBR-Systemen. Cloud-native-Anwendungen besitzen keine monolithische Anwendungsstruktur, sodass spezifische Ähnlichkeitsfunktionen bereitgestellt werden können, ohne eine Systemanpassung vornehmen zu müssen. Aktuell gibt es hierfür jedoch keine ontologiegestützten Vergleichsmaßstäbe, die verwendet werden können.

Die zuvor angeführte Gegenüberstellung offenbart vier wesentliche nicht-triviale Diskrepanzen:

- Es existiert keine Ontologie, die alle relevanten sprachlichen Ausdrucksmittel für sicherheitskritische IT-Projekte zur Verfügung stellt.
- Es existiert keine Implementierung eines ontologiegestützten CBR-Systems für sicherheitskritische IT-Projekte.
- Es existiert kein ontologiegestütztes Case-based Reasoning als Cloud-native-Anwendung für sicherheitskritische IT-Projekte inklusive der Implementierung ontologiegestützter Vergleichsmaßstäbe.
- Es existieren keine Vergleichsmaßstäbe für ontologiegestützte CBR-Systeme für sicherheitskritische IT-Projekte mittels Künstlicher Neuronaler Netze.

In problemorientierter Diktion lassen sich die zuvor genannten nicht-trivialen Diskrepanzen als folgende wissenschaftliche Probleme formulieren:

- *Ontologie-Problem:* Es liegt ein Transferproblem im Transfer der allgemeinen Technik der Ontologien auf die Domäne sicherheitskritischer IT-Projekte vor.
- *CBR-System-Problem:* Es liegt ein Transferproblem vor, ein allgemeines ontologiegestütztes CBR-System auf die Domäne sicherheitskritischer IT-Projekte zu transferieren.
- *Cloud-Native-Problem:* Es liegt ein Transferproblem beim Transfer der allgemeinen Techniken des Cloud-Native-Ansatzes und des ontologiegestützten Case-based Reasonings im Hinblick auf die Domäne sicherheitskritischer IT-Projekte vor.
- *Neuronales-Netz-Problem:* Es liegt ein Erkenntnisproblem vor, wie Künstliche Neuronale Netze zur Erstellung für ontologiegestützte CBR-Systeme für sicherheitskritische IT-Projekte genutzt werden können.

1.5 Intendierte Wissenschaftliche Erkenntnisse

Es ist nicht beabsichtigt, einen vollfunktionsfähigen Prototyp als Cloud-native-Anwendung zu implementieren. Die prototypische Entwicklung des Ähnlichkeitsalgorithmus und die Implementierung von spezifischen Ähnlichkeitsfunktionen dienen stattdessen „nur“ der Demonstration der Machbarkeit eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung. Zusammenfassend wird beabsichtigt, die folgenden wissenschaftlichen Erkenntnisse zu liefern:

- sicherheitskritische IT-Projekt-Ontologie
- CBR-System mit integrierter Ontologie für sicherheitskritische IT-Projekte
- sicherheitskritische IT-Projekte in Form von Projekten in einem CBR-System
- Ähnlichkeitsberechnung im CBR-System
- Ähnlichkeitsalgorithmus als Serverless-Funktion in einer Cloud-Umgebung
- Ähnlichkeitsfunktionen als Serverless-Funktionen in einer Cloud-Umgebung inklusive spezifischer Ähnlichkeitsfunktionen für die Verarbeitung von qualitativen Informationen aus sicherheitskritischen IT-Projekten unter der Verwendung von Künstlichen Neuronalen Netzen.

2 Grundlagen für ontologiegestütztes Case-based Reasoning für die Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

2.1 Ontologien

2.1.1 Einordnung des Ontologiebegriffs in einen informations- und betriebswirtschaftlichen Kontext

Der Begriff Ontologie entstammt ursprünglich der Philosophie und behandelt in diesem Kontext die Lehre des Seienden. In der Philosophie wird die Ontologie definiert als die Auseinandersetzung mit der begrifflichen Erfassung, der inhaltlichen Analyse sowie einer Charakterisierung des Ineinandergreifens der grundlegendsten Strukturen des Seienden. In der Philosophie wird mittels einer Ontologie der Versuch unternommen, die Grundstrukturen der Realität korrekt und allgemeingültig zu beschreiben.

Die einleitend umrissene, philosophische Nutzung des Begriffs Ontologie bildet nicht den Hintergrund desjenigen Begriffsverständnisses für Ontologien, das im Kontext des Wissensmanagements an Relevanz gewonnen hat. Hierzu ist eine Betrachtung des Ontologiebegriffs aus informationstechnischer Perspektive notwendig.

In der Informatik stellen Ontologien *sprachliche Ausdrucksmittel* für einen gemeinsamen Anwendungsbereich – eine „Domäne“ – dar. Sie werden als ein gemeinsam geteiltes gedankliches und sprachliches Verständnis dieser Domäne verstanden, das die Kommunikation zwischen menschlichen individuellen und kollektiven Akteuren (Personen, Personengruppen und Unternehmen) sowie artifiziellen Akteuren (Computern, „Maschinen“, Softwareprogrammen) unterstützt und dadurch den Austausch, das Teilen sowie das gemeinsame Anwenden von Wissen in Unternehmen erleichtert. Zu diesen Zwecken stellen Ontologien im Bereich der Informatik sprachliche Ausdrucksmittel für die Wissensrepräsentation zur Verfügung. Sie ermöglichen es insbesondere, die Bedeutung von natürlichsprachlich repräsentiertem, „qualitativem“ Wissen bei computergestützter Wissensspeicherung und -verarbeitung einzubeziehen.

Das Ziel einer Ontologie ist das Schaffen einer für die jeweiligen Kommunikationsteilnehmer gemeinsam geteilten Konzeptualisierung für einen präzise spezifizierten Realitätsausschnitt (Domäne). Dadurch kann mithilfe einer Ontologie das Wissen von relevanten Begriffen einer Domäne dokumentiert und transparent gemacht werden.

Zusammengefasst ist die Erwartung hinsichtlich des Einsatzes von Ontologien, ein systematisches Wissensmanagement zu stärken, indem Ontologien das Domänenwissen strukturieren. Dazu werden relevante Begriffe aus einer Domäne unter Berücksichtigung ihrer Semantik und ihrer Beziehungen zueinander in einer Ontologie spezifiziert. Die Repräsentation dieses Wissens erfolgt zumeist computergestützt durch den Einsatz von Ontologie-Editoren in computerverständlicher Form unter der Verwendung einer computerlesbaren Repräsentationssprache.

2.1.2 Definition von Ontologien

Eine allgemeine Definition von Ontologien in der Informatik existiert zurzeit nicht. Eine der am häufigsten genutzten Definitionen von Ontologien stammt von GRUBER, welcher eine Ontologie wie folgt definiert (GRUBER (1993), S. 199):

„An *ontology* is an explicit specification of a conceptualization."

Obwohl sich diese und artverwandte Definitionen großer Beliebtheit erfreuen, werden sie zugleich von diversen Autoren kritisiert, da insbesondere die Begriffe Konzeptualisierung und explizite Spezifikation undefiniert bleiben. Ein zusätzlicher Kritikpunkt ist, dass für Ontologien eingefordert wird, dass die Konstruktion dieser mit einem Prozess der Festlegung von Begriffen einhergeht. Diese Festlegung erfolgt mit mehreren Akteuren als geteiltes, gemeinsam akzeptiertes Begriffslexikon, da der Vorteil einer Ontologie gering wäre, wenn das gemeinsame Verständnis nicht vorliegt.

Um die genannten Kritikpunkte zu umgehen, wird für diesen Beitrag die Definition von ZELEWSKI als treffender erachtet. Diese lautet wie folgt (ZELEWSKI (2005), S. 153): „Eine Ontologie ist eine *explizite* und *formalsprachliche* Spezifikation derjenigen sprachlichen Ausdrucksmittel (für die Konstruktion repräsentationaler Modelle), die nach Maßgabe einer von *mehreren Akteuren gemeinsam verwendeten Konzeptualisierung* von realen Phänomenen, die in einem *subjekt- und zweckabhängig* eingegrenzten *Realitätsausschnitt* als *wahrnehmbar* oder *vorstellbar* gelten und für die *Kommunikation* zwischen den o. a. Akteuren benutzt oder benötigt werden, für „sinnvoll" erachtet werden."

ZELEWSKI definiert Ontologien insbesondere als eine Spezifizierung *sprachlicher Ausdrucksmittel*. Seine Definition wird aus den nachfolgenden drei Gründen als Arbeitsdefinition verwendet. Der erste Grund liegt in der *Sprachorientierung* dieser Definition. Die Sprachorientierung ist für die vorliegenden Untersuchungen von Bedeutung, weil Ontologien im Rahmen des ontologiegestützten Case-based Reasonings für die Bereitstellung sprachlicher Ausdrucksmittel zur Anwendung eines CBR-Systems verwendet werden. Der zweite Grund ist, dass in der Ontologiedefinition von ZELEWSKI zwei weitere Aspekte formuliert werden, welche in der Definition von GRUBER nicht enthalten sind, jedoch als wichtig erachtet werden. Zunächst ist der zu betrachtende Realitätsausschnitt zu identifizieren, auf dem die Wahrnehmung und Vorstellung basieren. Ein weiterer Aspekt betrifft die Konzeptualisierung eines Realitätsausschnitts, der sich auf relevante Phänomene beschränkt, die für die Kommunikation der Akteure von Bedeutung sind.

2.1.3 Ontologiekomponenten

Ontologien bestehen im Wesentlichen aus den folgenden Komponenten: Klassen (oder hier synonym verstanden: Konzepten), Relationen, Attributen, Restriktionen, Inferenzregeln und Integritätsregeln. Es ist jedoch nicht erforderlich, alle der sechs aufgeführten Bestandteile bei der Erstellung einer Ontologie zu verwenden.

Klassen stellen Begriffe zur sprachlichen Strukturierung eines Realitätsausschnitts dar. Eine Klasse umfasst eine Menge von Instanzen, die durch gemeinsame charakteristische Eigenschaften definiert sind und durch eine spezifische Bezeichnung repräsentiert werden. Klassen sind in einer taxonomischen Struktur angeordnet. Eine Taxonomie ordnet die Beziehungen zwischen den Klassen anhand der „is_a"-Relation. Der Vorteil einer Taxonomie ist, dass Vererbungsmechanismen anwendbar sind. Eine Unterklasse erbt die Eigenschaften ihrer Oberklasse.

Klassen können mithilfe von Relationen und auch Attributen weiter spezifiziert werden. Relationen und Attribute sind in ihren Grundfunktionen ähnlich, weswegen sie in der Fachliteratur häufig unter der Bezeichnung „Merkmale" oder „Eigenschaften" („properties") zusammengefasst werden. Mittels Relationen können Beziehungen zwischen Klassen in sowohl taxonomischer als auch nicht-taxonomischer Form hergestellt werden. Um taxonomische Beziehungen zwischen zwei Klassen herzustellen, wird die taxonomische Relation *„ist_ein"* verwendet. Alle weiteren Relationen werden als nicht-taxonomische Relationen bezeichnet. Nicht-taxonomische Relationen können frei gewählt werden. So können Klassen, die untereinander keine Ober-Unterordnungs-Beziehung haben, mittels nicht-taxonomischer Relationen in Beziehung zueinander gesetzt werden. Attribute werden einzelnen Klassen zugeordnet. Solche Attribute können verschiedene Datentypen wie numerische Werte oder Texte enthalten.

Instanzen sind eine konkrete Ausprägung einer Klasse. Instanzen verfügen exakt über die Attribute und Relationen der Klasse, zu der sie gehören, zusätzliche Attribute und Relationen können auf Instanzenebene nicht hinzugefügt werden. Vielmehr werden auf der Instanzenebene Instanzen konkrete Relations- und Attributswerte zugewiesen.

Sowohl Attribute als auch Relationen können mittels Restriktionen eingeschränkt werden. Die Verwendung von Restriktionen dient der Definition semantischer Einschränkungen von Klassen und deren Eigenschaften. Die Restriktionen werden mittels einer logischen deskriptiven Sprache ausgedrückt, um die Zuordnung von Attributen und Relationen zu Instanzen sowie die von Instanzen zu Klassen zu beschränken. Weitere Möglichkeiten zur Beschreibung von Zusammenhängen in einer Ontologie mittels einer logischen deskriptiven Sprache sind beispielsweise mengentheoretische Verknüpfungen von Klassen, numerische Quantifizierungen, die Hierarchisierung von Relationen sowie die Konstruktion von Inferenz- und Integritätsregeln. Inferenzregeln stellen einen wesentlichen Bestandteil wissensbasierter Systeme dar und dienen zur Schlussfolgerung neuen Wissens basierend auf bereits vorhandenem Wissen.

Die nachfolgende Abbildung 1 illustriert die zuvor genannten Ontologiekomponenten anhand eines Beispiels.

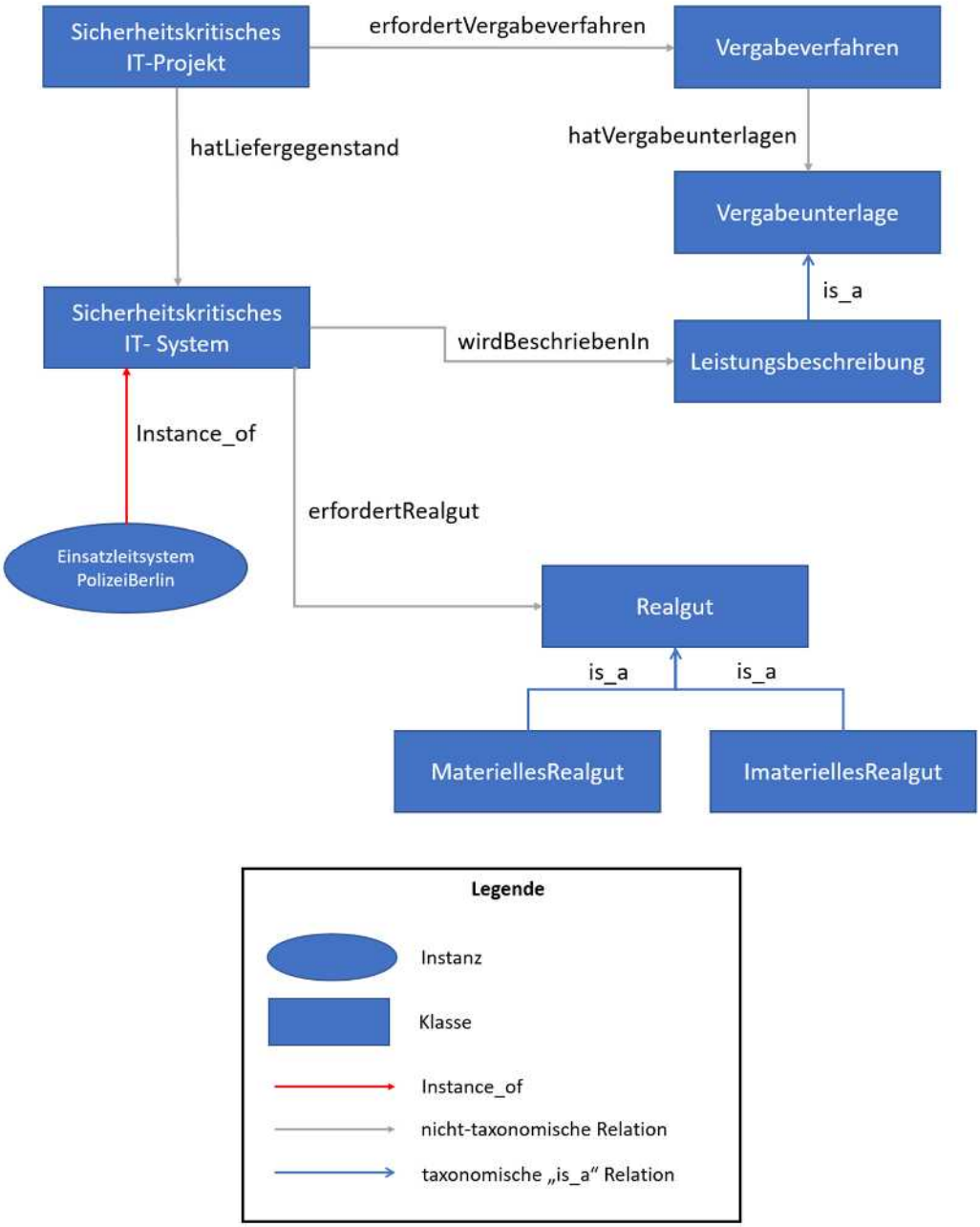


Abbildung 1: Beispiel-Ontologie zur Verdeutlichung der Ontologie-Komponenten

Anhand der Abbildung 1 wird exemplarisch dargestellt, dass die Klasse *Leistungsbeschreibung* eine Subklasse der Klasse *Vergabeunterlage* ist. Dies bedeutet, dass alle Instanzen, welche zur Instanzenmenge der Klasse *Leistungsbeschreibung* gehören, auch in der Instanzenmenge der Klasse *Vergabeunterlage* zusammengefasst sind. Der Inklusionszusammenhang gilt im Allgemeinen nur in einer Richtung. Daher wird in diesem Zusammenhang hinsichtlich der Relation *is_a* von einer Subsumptions-Relation gesprochen. Sie eignet sich generell dafür, eine Klasse (*Vergabeunterlage*, *Realgut*) einer anderen Klasse (*Leistungsbeschreibung*, *MateriellesRealgut* oder *ImmateriellesRealgut*) unterzuordnen. Die Beziehung zwischen den Klassen *sicherheits-*

kritisches IT-Projekt und *Vergabeverfahren* wird in der Abbildung 1 mittels der Relation *erfordertVergabeverfahren* ermöglicht. Die Klassen *sicherheitskritisches IT-Projekt* und *Vergabeverfahren* verfügen über Attribute, nämlich *benötigtSicherheitsüberprüfung* und *unterliegtGeheimhaltung*, die den Datentyp Boolean haben.

Eine mögliche Inferenzregel auf Basis des in Abbildung 1 dargestellten Beispiels ist:

Inferenzregel in SWRL	**natürlichsprachliche Übersetzung**
Vergabeverfahren(?vf) ^ unterliegtGeheimhaltung(?ivf, ?s) ^ swrlb:contains(?s, True" -> sicherheitskritischesIT-Projekt (?its) ^ benoetigtSicherheitsprüfung(?its, True)	Wenn ein Vergabeverfahren einer Geheimhaltungsstufe unterliegt, dann benötigt ein sicherheitskritisches IT-Projekt eine Sicherheitsprüfung.

Tabelle 1: Beispiel einer Inferenzregel

2.1.4 Ontologiebezogene Repräsentationssprachen

Repräsentationssprachen für Ontologien ermöglichen eine maschinenlesbare Darstellung der Komponenten einer Ontologie. Sie geben dadurch auch einen Gestaltungsspielraum für die Vokabularkontrolle und die formalen Begriffsdefinitionen vor. Nachfolgend wird ausschließlich die derzeit vorherrschende Web Ontology Language (OWL) als Repräsentationssprache vorgestellt. OWL ist eine vom W3C eingeführte Ontologiesprache, welche auf Resource Description Framework (Schema) RDF und RDFS aufbaut. OWL erweitert die Syntax von RDF bzw. RDFS um Beschreibungslogiken, wobei RDF/RDFS gewöhnlich in XML kodiert wird. In den Ausführungen von OWL lassen sich drei Varianten unterscheiden:

- OWL-Full ermöglicht die uneingeschränkte Nutzung aller OWL-Sprachelemente und enthält somit die vollständige Ausdruckskraft von OWL. Die hohe Ausdrucksstärke von OWL-Full hat den Nachteil, dass sie zu längeren Berechnungszeiten für Schlussfolgerungen führt.
- OWL-DL umfasst zwar die vollständigen OWL-Ausdrucksmittel. Aber die Einschränkungen zu OWL-Full zielen darauf ab, die Berechnungszeit der Schlussfolgerungen zu verringern.
- OWL Lite ist eine Teilsprache von OWL-DL und ermöglicht primär, Hierarchien von Klassen und einfache Restriktionen auszudrücken. Diese Teilsprache enthält nur die elementarsten Ausdrucksmittel.

2.2 Case-based Reasoning

2.2.1 Grundidee des Case-based Reasonings

Das Case-based Reasoning ist eine Technik aus der Erforschung der Künstlichen Intelligenz, die zur „intelligenten“ Lösung allgemeiner Probleme – oder synonym: Fälle oder Projekte – dient, die sich auf den Vergleich zwischen ähnlichen Problemen („Cases“) zurückführen lassen. Diese Technik stellt Gestaltungsgrundsätze für die „intelligente“ Wiederverwendung von Erfahrungswissen zur Verfügung, welche zur Entwicklung erfahrungsbasierter Systeme genutzt werden können.

Das Case-based Reasoning basiert auf zwei grundsätzlichen Annahmen, welche dem menschlichen Problemlösungsdenken nahekommen. Die erste Annahme ist, dass ähnliche Probleme auch ähnliche Lösungen haben. Die zweite Annahme ist, dass sich zwar jedes Problem voneinander unterscheidet, sich der Typ der Problemstellung jedoch wiederholt. Die Idee ist, dass die Problemlösungen in einer Datenbank gespeichert sind, um diese zur anschließenden Lösung von neuen Problemen jeweils in Abhängigkeit von ihrem ähnlichen Problemtyp nutzen zu können.

Ein Projekt – hier synonym für ein Problem oder einen Fall – umfasst aus der Perspektive des betriebswirtschaftlichen Projektmanagements stets drei Komponenten: eine Projektbeschreibung, eine Projektlösung und eine Bewertung der Projektlösung. Die Spezifizierung eines Projekts liegt in einer Projektwissensbasis vor, die oftmals auch als Fallbasis, Wissensbasis oder Projektdatenbank bezeichnet wird.

Der Begriff „Reasoning“ zielt darauf ab, anhand von alten, bereits durchgeführten Projekten Rückschlüsse auf ein neues, zu bewältigendes (z. B. ausgeschriebenes) Projekt zu ziehen. Dazu wird beim Case-based Reasoning ein neues Projekt – repräsentiert als ein „Fall“ – mit einer Sammlung von Projekten in der Projektwissensbasis verglichen, um möglichst ähnliche alte Projekte (Fälle) zu finden. Das ähnlichste alte Projekt wird als Ausgangspunkt für die Entwicklung eines Adaptionsverfahrens für das aktuelle Projekt verwendet. Nach der Bearbeitung wird das aktuelle Projekt mit seiner Projektlösung in die Projektwissensbasis übernommen. Die Projektwissensbasis nimmt durch jedes bearbeitete Projekt an Umfang zu. Daher bietet sie durch das stetige Anwachsen eine „breite“ Basis für die Suche nach einem ähnlichsten alten Projekt. Case-based Reasoning führt daher „automatisch“ zum Erlernen von Erfahrungswissen über bereits durchgeführte Projekte. Allerdings kann es auch passieren, dass kein Projekt in der Projektwissensbasis nützlich ist, also eine geforderte Mindestähnlichkeit zum neuen Projekt aufweist.

2.2.2 Case-based-Reasoning-Zyklus

Der CBR-Zyklus, der als typischer Ablauf des Case-based Reasonings gilt, ist auf die Ausführungen von AAMODT/PLAZA zurückzuführen; vgl. AAMODT/PLAZA (1994), S. 44-45. AAMODT/PLAZA beschreiben das Case-based Reasoning als einen zyklischen Prozess, der insgesamt vier Phasen umfasst; vgl. die nachfolgende Abbildung 2.

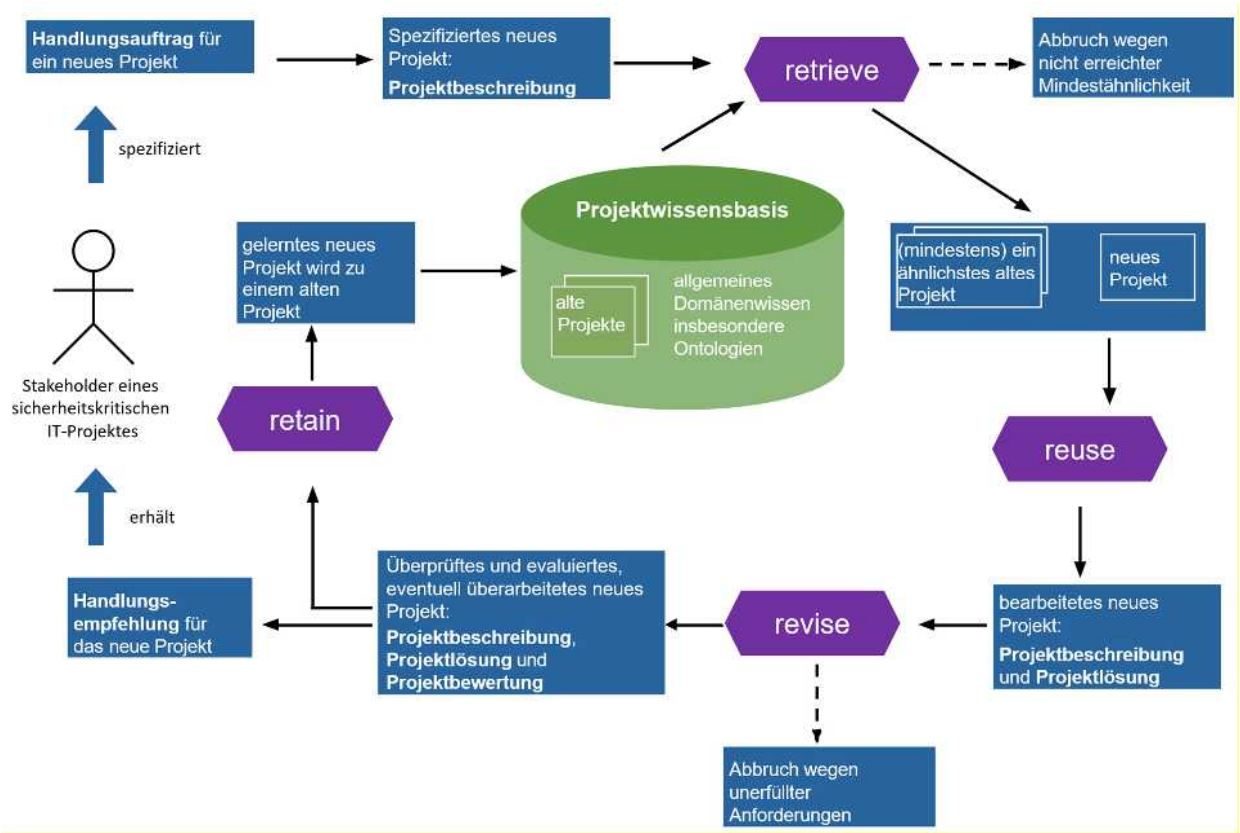


Abbildung 2: CBR-Zyklus für die Wissenswiederverwendung
im betrieblichen Projektmanagement

Wie in Abbildung 2 dargestellt, beginnt der CBR-Zyklus mit einem Handlungsauftrag für ein neues Projekt. Zur Lösung des neuen Projekts wird anhand der Projektbeschreibung in der Retrieve-Phase nach mindestens einem ähnlichsten alten, bereits bearbeiteten Projekt in der Projektwissensbasis gesucht. Sollte in der Retrieve-Phase kein altes Projekt gefunden werden, welches eine benutzerseitig vordefinierte Mindestähnlichkeit aufweist, dann wird die Anwendung der CBR-Technik ohne ein Ergebnis abgebrochen.

In der Reuse-Phase erfolgt eine Analyse der Abweichung der Projektbeschreibungen zwischen dem ermittelten ähnlichsten alten Projekt und dem angefragten neuen Projekt. Anhand der zuvor identifizierten Abweichungen soll die Lösung des ähnlichen alten Projekts an die Beschreibung des neuen Projekts angepasst werden. Das Ergebnis dieser Phase ist eine neue Projektlösung.

Die darauffolgende Revise-Phase dient der Überprüfung der neuen Projektlösung und ihrer Eignung zur Wiederverwendung. Beispielsweise kann die vorläufige Projektlösung sowohl manuell durch erfahrene Projektmanager als auch computergestützt, beispielsweise mittels Integritätsregeln, hinsichtlich ihrer Eignung beurteilt werden. Sollte die bisher erarbeitete Projektlösung nicht vollständig plausibel erscheinen, können Korrekturen an der vorliegenden Projektlösung vorgenommen werden. Ziel der vorgenommenen Korrekturen ist es, die vorläufige Projektlösung so zu überarbeiten, dass alle Anforderungen an eine plausible und wiederverwendbare Projektlösung erfüllt werden. Sollte sich die Projektlösung aufgrund von Plausibilitäts- oder Wiederverwendbarkeitsanforderungen nicht erfüllen lassen, erfolgt ein Abbruch wegen „irreparabler Lösungsanforderungen“. Ansonsten fließen die gewonnenen Erkenntnisse der Revise-Phase in die Projektbewertung ein.

Zuletzt, in der Retain-Phase, wird das Triple aus der Projektbeschreibung, der Projektlösung und der Projektbewertung für das neue Projekt als Wissen in die Projektwissensbasis aufgenommen. Parallel zur Retain-Phase wird die ermittelte Projektlösung und optional auch die Projektbewertung als Handlungsempfehlung für das neue Projekt an das Projektmanagement herausgegeben.

2.3 Projektmanagement von sicherheitskritischen IT-Projekten

2.3.1 Projektmanagement

Für das Management von sicherheitskritischen IT-Projekten werden in der Regel Methoden des Projektmanagements angewendet.

Das Projektmanagement wird definiert als die Gesamtheit von Führungsaufgaben, -organisationsformen, -techniken und -mitteln für die Initiierung, Definition, Planung, Steuerung und den Abschluss von Projekten. Es umschreibt die Anwendung von Wissen, Fertigkeiten und Techniken auf Projektaktivitäten, um Projektanforderungen zu erfüllen. Die Projektanforderungen sollen innerhalb der Leistungsziele Zeit, Kosten, Qualität, Umfang, Nutzen und Risiko erfüllt werden.

Der häufig genannte Vorteil eines Projektmanagementstandards ist die Nutzung einer gemeinsamen Sprache und somit eines gemeinsamen Verständnisses einer Projektbearbeitung, um die Reibungsverluste zwischen den beteiligten Personen im Projekt zu reduzieren. In den letzten Jahren haben sich verschiedene Projektmanagementstandards etabliert, die sich auch für die Durchführung von sicherheitskritischen IT-Projekten eignen. Die gängigsten Projektmanagementstandards sind International Project Management Association (IPMA), Project Management Institute (PMI), Projects in Controlled Environments (PRINCE2) und Scrum.

Die einschlägige Fachliteratur unterscheidet häufig zwischen einerseits klassischen Projektmanagementmethoden, wie z. B. IPMA, PMI und PRINCE2, sowie andererseits agilen Projektmanagementmethoden, wie z. B. Scrum. Es ist jedoch zu beachten, dass sich klassische und agile Projektmanagementmethoden nicht grundsätzlich widersprechen müssen, sondern sich

auch gegenseitig ergänzen können. Daher sind auch hybride Ansätze möglich; vgl. HILMER/ KRIEG (2014), S. 47-48; HABERMANN (2013), S. 93-94. Die Auswahl einer Projektmanagementmethode hängt im Wesentlichen vom Vorhaben, der Arbeitsweise eines Auftragnehmers und den Anforderungen eines Auftraggebers ab. In den nachfolgenden Erläuterungen zu PRINCE2 und Scrum wird die „Wiederverwendung von Erfahrungswissen" aus den beiden Projektmanagementmethoden herausgestellt. Die Fokussierung auf diese zwei genannten Projektmanagementmethoden basiert auf folgenden Überlegungen:

- PRINCE2 gilt als weltweit meistverwendete Projektmanagementmethode neben IPMA und PMI. Vgl. ERNE (2019), S. 20. In PRINCE2 sind „Wissen" sowie „Lernen aus Erfahrungen" wichtige Bestandteile.
- In dem agilen Standard „Scrum" stellt die „Sprint Retrospective" einen zentralen Punkt dar, um aus Erfahrungen zu lernen.
- Die agile Vorgehensweise gewinnt im Projektmanagement sicherheitskritischer IT-Projekte zunehmend an Bedeutung. Ein Beispiel ist das Unternehmen Eurofunk Kappacher GmbH, das sicherheitskritische IT-Projekte durchführt und die agile Vorgehensweise als bevorzugten Standard bezeichnet, wenn es darum geht, Einsatzleitsysteme zu implementieren. In einigen Vergabeverfahren wird von den Auftraggebern explizit die Anwendung der Scrum-Projektmanagementmethode gefordert, um ein sicherheitskritisches IT-Projekt umzusetzen. Als Beispiel wird die Implementierung eines Einsatzleitsystems für die Polizei im Land Brandenburg genannt.
- Die Projektmanagementmethoden PRINCE2 und Scrum lassen sich zusammen als hybrider Ansatz in einem sicherheitskritischen IT-Projekt anwenden.
- Es existiert bereits eine PRINCE2- und Risikomanagement-Ontologie, die für sicherheitskritische IT-Projekte ausgelegt ist und als Grundlage für die Konstruktion einer Ontologie für sicherheitskritische IT-Projekte genutzt werden kann. Die PRINCE2- und Risikomanagement-Ontologie wird in WEBER et al. (2021), S. 12-23, beschrieben und hier wiederverwendet.

Aufgrund der beschriebenen Begründungen erscheint es sinnvoll, den wissensbasierten Fokus von PRINCE2 und Scrum mit dem praktischen Ansatz des Case-based Reasonings für die Wiederverwendung von Erfahrungswissen aus sicherheitskritischen IT-Projekten zu verbinden.

2.3.2 Projektmanagementdomäne: Sicherheitskritische IT-Projekte

Eine einheitliche Definition von sicherheitskritischen IT-Projekten existiert zurzeit nicht. Insbesondere ist zu beachten, dass die Definition eines sicherheitskritischen IT-Projekts je nach Branche und Unternehmensumfeld variieren kann. Unabhängig von der Branche ist es jedoch notwendig, eine Risikoanalyse durchzuführen, um sicherzustellen, dass die richtigen Maßnahmen ergriffen werden, um die IT-Systeme und -Daten zu schützen, die durch ein IT-Projekt

entwickelt und bereitgestellt werden sollen. In diesem Beitrag liegt der Fokus auf *öffentlichen Projekten*, die durch öffentliche Mittel finanziert und durch eine öffentliche Vergabe vergeben werden.

Um ein sicherheitskritisches IT-Projekt zu definieren, müssen zunächst die Begriffe „Projekt", „IT-Projekt" und „sicherheitskritisch" definiert werden.

PRINCE2 definiert ein Projekt als „eine für einen befristeten Zeitraum geschaffene Organisation, die den Auftrag hat, mindestens ein Produkt entsprechend einem vereinbarten Business Case zu liefern" (AXELOS (2018), S. 8). Ein Projekt hat demnach folgende Merkmale (vgl. AXELOS (2018), S. 8-9):

- Veränderung: Ein Projekt soll eine Veränderung realisieren.
- befristet: Ein Projekt ist ein befristetes Vorhaben.
- bereichsübergreifend: Die Arbeit in einem Projekt erfolgt mit einem Team von Personen mit unterschiedlichen Fähigkeiten.
- einzigartig: Das Vorhaben eines Projektes ist einzigartig.
- unsicherheitsbehaftet: Das Projekt bringt Chancen, jedoch auch Bedrohungen in Form von Risiken mit sich.

Ein IT-Projekt hat die gleichen Merkmale, wie der zuvor erläuterte Projektbegriff. Jedoch befasst sich ein IT-Projekt konkret mit der Implementierung eines informationstechnischen Systems.

Sicherheitskritische IT-Projekte weisen alle zuvor genannten Merkmale eines IT-Projekts auf und befassen sich darüber hinaus mit der Implementierung eines sicherheitskritischen IT-Systems. Sicherheitskritische IT-Systeme (auch: Mission Critical Systems) finden sich in Kritischen-Infrastruktur-Organisationen wieder und erfordern eine „Funktionale Sicherheit"; vgl. die internationale Norm IEC 61508. Die Bundesregierung definiert Kritische-Infrastruktur-Organisationen als „Organisationen oder Einrichtungen mit wichtiger Bedeutung für das staatliche Gemeinwesen, bei deren Ausfall oder Beeinträchtigung nachhaltig wirkende Versorgungsengpässe, erhebliche Störungen der öffentlichen Sicherheit oder andere dramatische Folgen eintreten würden." (BUNDESMINISTERIUM DES INNERN (2009), S. 3). In Deutschland werden Organisationen und Einrichtungen aus den Bereichen Energieversorgung, Informationstechnik und Telekommunikation, Transport und Verkehr, Gesundheit, Wasser, Ernährung, Finanz- und Versicherungswesen, Staat und Verwaltung sowie Medien und Kultur zu den kritischen Infrastrukturen gezählt.

Ein Projektfehlschlag oder eine IT-Störung im späteren Betrieb eines sicherheitskritischen IT-Systems, was auf eine mangelnde Arbeit im sicherheitskritischen IT-Projekt hindeutet, hätten eine weitreichende Auswirkung. Die frühe Identifikation und Bewertung von Risiken stellen aufgrund der drohenden Konsequenzen eine sehr wichtige Aufgabe innerhalb eines sicher-

heitskritischen IT-Projekts dar. Die Identifikation und Bewertung von Risiken erfolgt kontinuierlich und unabhängig von der jeweiligen Projektphase. Aufgrund der Risiken und ihrer drohenden Auswirkungen sind sicherheitskritische IT-Projekte im Projektmanagement anders zu führen als normale IT-Projekte; vgl. GASSMANN (2001), S. 12.

Der Begriff des Risikos nimmt eine zentrale Stellung innerhalb eines sicherheitskritischen IT-Projekts ein und bedarf einer besonderen Form des Risikomanagements im sicherheitskritischen IT-Projektmanagement. Im Risikomanagement eines sicherheitskritischen IT-Projekts müssen potenzielle Risiken als „schwache Signale“ frühzeitig identifiziert und korrekt interpretiert werden, um die drohenden Auswirkungen einzuschätzen und eventuell erforderliche Gegenmaßnahmen vorausschauend zu planen.

Zusammenfassend wird in diesem Beitrag ein sicherheitskritisches IT-Projekt als ein IT-Projekt verstanden, das folgende spezifische Eigenschaften aufweist:

- Ein sicherheitskritisches IT-Projekt ist ein Projekt, bei dem die Integrität, Vertraulichkeit und Verfügbarkeit des zu implementierenden sicherheitskritischen IT-Systems von zentraler Bedeutung sind und dessen Schutzbedarfskategorie daher als „sehr hoch“ eingestuft wird.
- Ein sicherheitskritisches IT-Projekt erfordert eine Funktionale Sicherheit, um ein hohes Maß an Schutz vor drohenden Schäden durch Fehlfunktionen oder unerwünschtes Verhalten des sicherheitskritischen IT-Systems zu gewährleisten.
- Aufgrund der Feststellung als kritische Infrastruktur und der damit verbundenen gesellschaftlichen Abhängigkeit von dem IT-System treffen gesetzliche Anforderungen zu.
- Einer der zentralen Stakeholder des Projektes ist eine KRITIS-Organisation.
- Das Projekt unterliegt einem überdurchschnittlich hohen Risiko.
- Ein Ausfall oder ein Datenverlust im zu implementierenden sicherheitskritischen IT-System kann zu schwerwiegenden Auswirkungen auf die gesellschaftliche Ordnung führen.
- Durch ein vorgelagertes öffentliches Vergabeverfahren wird ein Auftrag für ein sicherheitskritisches IT-Projekt an einen Bieter vergeben.

Aufgrund der genannten Eigenschaften sind eine besondere Form des Risikomanagements und eine damit verbundene frühzeitige Identifizierung von Risiken in Form einer Interpretation von „schwachen Signalen“ notwendig.

2.4 IT-Anwendungen

2.4.1 Monolithische Anwendungen

Der Begriff „monolithische Anwendung“ ist in der Informationstechnologie nicht einheitlich definiert. Jedoch wird unter einem „Monolithen“ eine IT-Anwendung (Software) verstanden, deren funktionalen Elemente nicht voneinander getrennt sind und als ein Codeblock bereitgestellt werden. Die Softwareanwendung ist in sich geschlossen, unabhängig von anderen Anwendungen und wird in der Regel in Zwei-Schicht- oder Drei-Schicht-Architekturen klassifiziert. Die Drei-Schicht-Architektur unterscheidet zwischen der Datenhaltungsebene, der Fachkonzeptebene und der Präsentationsebene. Bei einer Zwei-Schicht-Architektur entfällt die Fachkonzeptebene.

Eine monolithische Anwendung zeichnet sich durch folgende Merkmale aus:

- Jegliche Funktionalitäten sowie Komponenten einer monolithischen Anwendung werden traditionell in einer einzigen Codebasis implementiert.
- Eine monolithische Anwendung nutzt in der Regel eine Datenbank, um Daten zu speichern sowie abzurufen.
- Eine Weiterentwicklung der Applikationsfunktion in einer monolithischen Anwendung wird in der Regel schrittweise vorgenommen.
- Monolithische Anwendungen können in einem betrieblichen Umfeld schwieriger skaliert werden, da sie auf einer Codebasis aufgebaut sind und nicht horizontal skaliert werden können.
- Monolithische Anwendungen benötigen in der Praxis häufig eine längere Entwicklungszeit, da alle Applikationsfunktionalitäten in einer einzigen Codebasis implementiert werden und als Gesamtanwendung getestet werden müssen.

2.4.2 Cloud-native-Anwendungen

Die Bezeichnung „Cloud-native-Anwendung“ (auch: „Cloud-native-Computing“) ist nicht einheitlich definiert. Besonders hervorzuheben ist die (im Original deutschsprachige) Definition von KRATZKE (2022), S. 35, die eine Cloud-native-Anwendung definiert als: „ein verteiltes, beobachtbares, elastisches und auf horizontale Skalierbarkeit optimiertes Service-of-Services System, das seinen Zustand in (einem Minimum an) zustandsbehafteten Komponenten isoliert. In vereinfachter Form bedeutet dies, dass die Softwareanwendung gemäß designorientierten Prinzipien für die Cloud entworfen wird und anschließend in einer Cloud-Umgebung bereitgestellt und ausgeführt wird. Folgende grundlegende Merkmale werden einer Cloud-nativen-Anwendung zugeschrieben (vgl. VETTOR/SMITH (2023), S. 5-10; CALDATO (2020), S. 1-2):

- Die Softwareanwendung wird ausschließlich in einer Cloud entwickelt und ist für die ausschließliche Nutzung in der Cloud ausgelegt.
- Cloud-native-Anwendungen nutzen Open-Source-Technologien und sind in erster Linie auf Transparenz und Interoperabilität ausgerichtet.
- Der Cloud-native-Ansatz konzentriert sich auf das Erstellen von Funktionalitäten, die auf einer Serverlosigkeit („Serverless“) basieren und als Microservices gekapselt bereitgestellt werden können.
- Cloud-native-Anwendungen sind für die horizontale Skalierung ausgelegt.

In aktuellen Studien werden sowohl die Cloud-nativen-Anwendungen als auch die Entwicklung dieser Anwendungen, die ebenfalls in einer Cloud erfolgt, als State of the Art angesehen; vgl. DELOITTE (2022), S. 28-34; SLASHDATA (2021), S. 13-16; CAPGEMINI (2021), S. 36; GARTNER (2021), S. 7; LÜNENDONK (2021), S. 7-8 u. 13-36, insbesondere S. 17-36. Diese Entwicklung wird maßgeblich durch die Entwicklung von Cloud-Technologien vorangetrieben.

In diesem Beitrag werden für die Cloud-native-Anwendung ein exemplarischer „Klick-Prototyp“ bereitgestellt sowie diverse Serverless-Funktionen implementiert. Die Hauptfunktion verarbeitet das Ergebnis und gibt es als Antwort auf die Anfrage an das API-Gateway zurück, welches es wiederum an den angefragten Client weiterleitet. Architektonisch lässt sich das zuvor dargestellte Beispiel in einer Amazon Web Service (AWS) Cloud wie folgt darstellen.

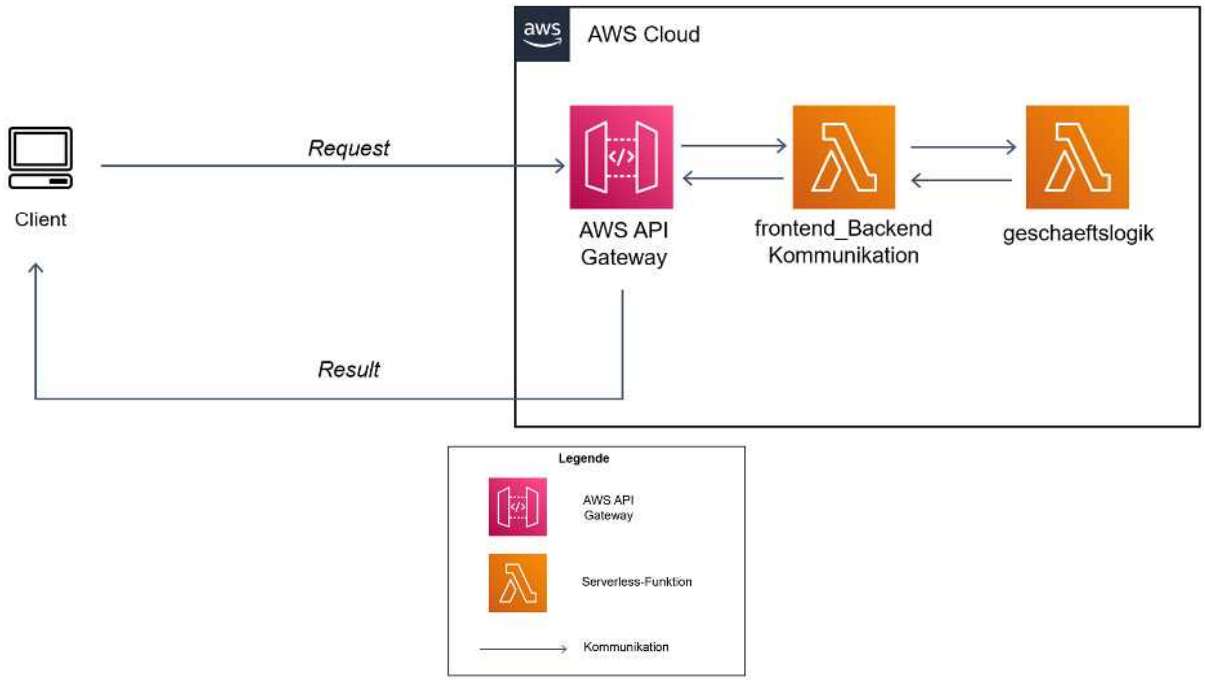


Abbildung 3: Aufruf einer Serverless-Funktion in der Amazon Web Service (AWS) Cloud

3 Anwendung des ontologiegestützten Case-based Reasonings für die Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

3.1 Vorgehensweise für die Anwendung des ontologiegestützten Case-based Reasonings für die Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

Für die Anwendung des ontologiegestützten Case-based Reasonings für sicherheitskritische IT-Projekte sind fünf Schritte hilfreich, die aufeinander aufbauen (die Begriffe „Projekt“ und „Fall“ werden in diesem Beitrag synonym verwendet):

1. Konstruktion einer sicherheitskritischen IT-Projekt-Ontologie,
2. Integration der sicherheitskritischen IT-Projekt-Ontologie in das CBR-Tool jCORA,
3. Beschreibung und Modellierung von Projekten zur Repräsentation von Praxisbeispielen,
4. Ähnlichkeitsberechnung zwischen Projekten mithilfe des ontologiegestützten CBR-Tools jCORA,
5. Anpassung der Lösung (mindestens) eines ähnlichsten alten Projekts an ein neues Projekt als ein Anschlussproblem, das in diesem Beitrag nur skizziert, aber nicht intensiv behandelt wird, sondern weiterführender Forschungsarbeiten bedarf.

Die genannten Schritte werden in den nachfolgenden Kapiteln näher beschrieben und dienen als Vorgehensweise für die Anwendung des ontologiegestützten Case-based Reasonings für sicherheitskritische IT-Projekte.

3.2 Konstruktion einer sicherheitskritischen IT-Projekt-Ontologie

3.2.1 Auswahl einer Konstruktionsmethode für die Entwicklung einer sicherheitskritischen IT-Projekt-Ontologie

In der Fachliteratur existieren verschiedene Konstruktionsmethoden für die Erstellung einer Ontologie; vgl. GÓMEZ-PÉREZ/FERNÁNDEZ-LÓPEZ/CORCHO (2004), S. 113-154. Eine Standardvorgehensweise, um eine Ontologie zu erstellen, existiert jedoch nicht.

In diesem Beitrag wird die Ontologiekonstruktion an die Konstruktionsmethode von NOY/MCGUINNESS angelehnt; vgl. NOY/MCGUINNESS (2001), S. 4-23. Die Autoren schlagen für die Konstruktion sieben nacheinander folgende Aktivitäten vor, die in diesem Beitrag um eine weitere Aktivität – die Definition von Regeln (Semantic Web Rules) – erweitert werden. Diese Erweiterung beruht darauf, dass für die sicherheitskritische IT-Projekt-Ontologie neben Kardinalitäten auch Semantic Web Rules definiert werden, die weitergehende Schlussfolgerungen

(Reasoning) ermöglichen. Eine solche Regelentwicklung sehen NOY/MCGUINNESS in ihrer Konstruktionsmethode jedoch nicht vor.

Die für diesen Beitrag relevanten 8 Aktivitäten für die Konstruktion einer Ontologie lauten wie folgt:

1. Festlegung des Anwendungsbereichs (Domäne) der Ontologie
2. Prüfung bestehender Ontologien für die Nutzung einer neuen Ontologie
3. Bestimmung wichtiger Begriffe für die Ontologie
4. Festlegen der Klassen und der taxonomischen Klassenstruktur
5. Definition der Eigenschaften der Klassen
6. Deklaration der Eigenschaften mittels Kardinalitäten
7. Definition von Semantic Web Rules
8. Erstellung der globalen Instanzen

Bei der praktischen Umsetzung des voranstehenden Vorgehensschemas kann es dazu kommen, dass eine Aktivität wiederholt werden muss. Die „lineare“, aufeinander folgende Umsetzung der Aktivitäten ist zuweilen schwer möglich, da während der Konstruktion einer Ontologie oftmals eine Anpassung von vorherigen Aktivitäten erforderlich ist. Daher sind die o. a. acht Aktivitäten eher als eine idealisierte Vorgehensweise für die Ontologiekonstruktion zu verstehen und können neben einem mehrmaligen Durchlaufen einer Aktivität auch weitere Teilaktivitäten innerhalb einer Aktivität beinhalten.

Des Weiteren ist es wichtig zu verstehen, dass die zu erstellende Ontologie ein Basisprodukt in Form einer OWL-Datei ist und keine eigenständige Softwareanwendung darstellt. Eine Ontologie lässt sich zu einem späteren Zeitpunkt erweitern oder andersartig modifizieren. Dies kann notwendig sein, wenn sich die Benutzersicht ändert oder weitergehende Aspekte, wie z. B. „radikal“ neuartige Projekte, berücksichtigt werden sollen.

3.2.2 Auswahl von Protégé als Ontologieeditor

Für die Konstruktion einer Ontologie stehen diverse Ontologieeditoren zur Verfügung. Die in der Literatur häufig genannten Ontologieeditoren sind z. B. Apollo, OntoStudios, Swoop und Protégé. Eine zwingende Verwendung eines Ontologieeditors besteht zwar nicht. Für die Konstruktion von umfangreichen Ontologien bietet sich jedoch im Allgemeinen die Verwendung an. Ontologieeditoren bieten toolgestützte Konstruktionshilfen an und ermöglichen die Validierung der konstruierten Ontologie mittels eines Reasoners. Des Weiteren existieren umfangreiche dokumentierte Hilfestellungen, um die Nutzung eines Ontologieeditors zu erleichtern.

Zur Entwicklung der sicherheitskritischen IT-Projekt-Ontologie wurde der Ontologieeditor Protégé in der Version 5.5.0 ausgewählt. Die Entscheidung für den Einsatz von Protégé als Ontologieeditor wird durch die folgenden Argumente begründet:

- In mehreren Untersuchungen, wie z. B. von BEIßEL (2011), S. 85-112, werden verschiedene Ontologieeditoren miteinander verglichen. Der Ontologieeditor Protégé erweist sich mehrfach als empfehlenswerte Alternative. In der sehr detaillierten Untersuchung von BEIßEL wurden Ontologieeditoren im Hinblick auf ihre Funktionalität, Zuverlässigkeit und Benutzbarkeit beurteilt.
- Es existieren verschiedene Quellen sowie Onlinedokumentationen in Form von Videoaufzeichnungen, um sich als potenzieller Nutzer mit der Software vertraut zu machen. Für den Ontologieeditor Protégé liegt zudem eine Online-Hilfe vor.
- Es lassen sich verschiedene Module als „Plug-Ins“ installieren. Die Erweiterbarkeit reicht von zusätzlichen Visualisierungsmöglichkeiten bis hin zu zusätzlichen Reasoning-Komponenten.

Es lassen sich jedoch auch Nachteile feststellen, die für diesen Beitrag zunächst als akzeptabel angesehen werden:

- In dem Ontologieeditor sind Fehler enthalten, die dazu führen, dass das Programm zuweilen abstürzt.
- Die Usability von Protégé wird als verbesserungswürdig angesehen, da eine intuitive Nutzung des Tools – ohne vorherige Sichtung von Unterlagen – schwer möglich ist.
- Es handelt sich um ein Desktop-basiertes System mit der Notwendigkeit einer Java-Runtime-Umgebung.

Protégé ist ein Ontologieeditor, der an der Stanford University entwickelt wurde. Der Ontologieeditor unterliegt aktuell der General Public License und steht inklusive Quellcode frei zur Verfügung. Protégé unterstützt zwei Arten der Ontologieerstellung, und zwar einen Frame-basierten sowie einen OWL-basierten Ansatz. Der Frame-basierte Ansatz ermöglicht es, klassische Komponenten einer Ontologie wie Klassen und Relationen zu erstellen. Der OWL-basierte Ansatz, welcher in diesem Beitrag genutzt wird, ermöglicht es, die Ausdrucksstärke von OWL und RDF/RDFS vollumfänglich zu nutzen.

Protégé bietet die Möglichkeit, seinen Funktionsumfang durch Plug-ins zu erweitern. Für diesen Beitrag sind folgende Plug-ins relevant:

- OntoGraf
- OWLViz
- SWRL-Tab

Die beiden Plug-ins „OntoGraf“ sowie „OWLViz“ ermöglichen zusätzliche Visualisierungen einer Ontologie. Das Plug-in „SWRL-Tab“ dient zur Unterstützung für die Konstruktion von Regeln.

Neben den bereits genannten Komponenten sind die in Protégé enthaltenen integrierten Inferenzmechanismen, die sogenannten Reasoner, besonders zu erwähnen. Beispielsweise zu erwähnen sind Fact++, Pellet oder HermiT. Mittels solcher Reasoner lässt sich implizites Wissen ableiten. Darüber hinaus ermöglichen die Reasoner, die logische Konsistenz einer Ontologie zu überprüfen.

3.2.3 Anwendung der Konstruktionsmethode für die Konstruktion einer sicherheitskritischen IT-Projekt-Ontologie

3.2.3.1 Festlegung des Anwendungsbereichs einer IT-Projekt-Ontologie

Der abzudeckende Ausschnitt des Anwendungsbereichs (Domäne) einer IT-Projekt-Ontologie wird definiert. Es soll festgelegt werden, welche Aspekte eines Anwendungsbereichs in der Ontologie beschrieben werden sollen und welche Bereiche zu vernachlässigen sind. Darüber hinaus sollen in diesen Schritten der Detaillierungsgrad und der Anwendungszweck der Ontologie festgelegt werden.

Um den Anwendungsbereich festzulegen, werden zunächst Grundfragen formuliert. Die nachfolgende Tabelle 2 zeigt die Grundfragen an die zu erstellende sicherheitskritische IT-Projekt-Ontologie.

Grundfragen	**Antworten**
Auf welche Domäne bezieht sich die Ontologie?	Die Ontologie bezieht sich auf die Anwendungsdomäne sicherheitskritische IT-Projekte.
Für welchen Zweck wird die Ontologie verwendet?	Der Zweck der Ontologie bezieht sich auf die Strukturierung und Darstellung von domänenspezifischem Wissen für sicherheitskritische IT-Projekte. Die Ontologie dient als Spezifikation von gemeinsam verwendeten sprachlichen Ausdrucksmitteln, um die Kommunikation zwischen Akteuren zu verbessern, die in sicherheitskritischen IT-Projekten zusammenarbeiten. Die Ontologie dient als Grundlage für das prototypische CBR-Tool jCORA sowie für die Verwendung in einem Cloud-basierten, ontologiegestützten CBR-System, um die Wiederverwendung von Erfahrungswissen aus sicherheitskritischen IT-Projekten zu unterstützen.

Wer soll die Ontologie benutzten?	Die sicherheitskritische IT-Projekt-Ontologie soll von allen Akteuren genutzt werden, die im Bereich sicherheitskritischer IT-Projekte tätig sind. Dies schließt insbesondere öffentliche Behörden und Organisationen mit Sicherheitsaufgaben ein, die sicherheitskritische IT-Projekte vergeben, sowie Auftragnehmer, die solche Projekte durchführen.

Tabelle 2: Grundfragen zur Eingrenzung des Anwendungsbereichs

Die Grundfragen setzen zwar den groben Rahmen, reichen jedoch nicht allein aus, um den Anwendungsbereich zu konkretisieren. Für die Konkretisierung wird oftmals die Formulierung von Kompetenzfragen („Competency Questions“) vorgeschlagen. Das Ziel der Formulierung der Kompetenzfragen ist es, dass der Ontologiekonstrukteur den Fokus bei der Spezifizierung der Ontologie auf die Beantwortung der Kompetenzfragen setzen soll. Die Beantwortung der Kompetenzfragen stellt aus der Perspektive des Ontologiekonstrukteurs die Bereiche dar, die von besonderem Interesse sind. Darüber hinaus sollen die Kompetenzfragen als zu erwartende Anfragen von potentiellen Nutzern an eine Ontologie verstanden werden. Es ist wünschenswert, die Kompetenzfragen mithilfe der in der sicherheitskritischen IT-Projekt-Ontologie bereitgestellten sprachlichen Ausdrucksmittel vollumfänglich formulieren zu können. Es ist wünschenswert, eine möglichst vollständige Liste dieser Kompetenzfragen zu entwickeln, da dadurch eine Prüfung auf Abdeckung der relevanten Aspekte sowie auf Korrektheit und Vollständigkeit der Ontologie möglich ist.

Um praxisnahe Kompetenzfragen zu formulieren, wurden sechs Experten befragt, die in sicherheitskritischen IT-Projekten in unterschiedlichen Rollen im Umfeld von sicherheitskritischen IT-Projekten arbeiten. Diese Experten sind Projektmanager, die nach ihren Angaben jeweils mehr als fünf Jahre Projekterfahrung mit einem Projektvolumen von insgesamt mehr als 5 Mio. Euro in der Integration von sicherheitskritischen IT-Systemen besitzen. Als Ergebnisse sind die nachfolgend dargestellten Kompetenzfragen zu nennen:

- Welche Risikotypen existieren bei sicherheitskritischen IT-Projekten?
- Wer sind die Stakeholder eines sicherheitskritischen IT-Projekts?
- Welche Eigenschaften definieren die Verfügbarkeit eines sicherheitskritischen IT-Systems?
- Wie wird die Fehlertoleranz eines sicherheitskritischen IT-Systems gemessen?
- Wie sieht der Aufbau (Systemumgebungen) eines sicherheitskritischen IT-Systems aus?
- Welchen Einfluss haben Gesetzesänderungen bei sicherheitskritischen IT-Projekten?
- Welche Kompetenzen sind bei der Durchführung von sicherheitskritischen IT-Projekten notwendig?

- Wie sieht ein Lastenheft (definierter Lieferumfang) bei sicherheitskritischen IT-Projekten aus?
- Wie sieht die Abgrenzung/Einbettung/Wechselwirkung zu anderen sicherheitskritischen IT-Projekten und -Umgebungen aus?
- Wie sieht die notwendige erhöhte Qualitätssicherung bei sicherheitskritischen IT-Projekten aus?
- Welche Methoden können zur Risikobeurteilung eingesetzt werden?
- Welche Sicherheitsstufen existieren bei sicherheitskritischen IT-Projekten?
- Welche KRITIS-Sektoren können sicherheitskritische IT-Systeme besitzen?
- Welche Verfügbarkeitsklassen existieren bei sicherheitskritischen IT-Systemen?
- Welche Risikomaßnahmen existieren bei sicherheitskritischen IT-Projekten?
- Wie erfolgt die vorausschauende Betrachtung des Stands der Technik (aktuell, zukünftig) insbesondere bei langlaufenden IT-Projekten?
- Welche Schutzmaßnahmen existieren bei sicherheitskritischen IT-Systemen?
- Welche Einflussfaktoren existieren bei sicherheitskritischen IT-Projekten?
- Wie sieht eine Projektorganisation bei sicherheitskritischen IT-Projekten aus?
- Was unterscheidet ein „normales“ IT-Projekt von einem sicherheitskritischen IT-Projekt?
- Welche Dokumente sind für die Nachvollziehbarkeit eines sicherheitskritischen IT-Systems notwendig?
- Welche Komponenten sind für die Nachweisbarkeit eines sicherheitskritischen IT-Projekts und sicherheitskritischen IT-Systems von Relevanz?
- Wo und mit welchen Eigenschaften wird die Bedienbarkeit eines sicherheitskritischen IT-Systems gemessen?
- Welche Eigenschaften hat ein sicherheitskritisches IT-System?
- Welche Umsetzungsstrategien existieren in sicherheitskritischen IT-Projekten?
- Welche Auswirkung hat das Ausschreibungsverfahren für die spätere Realisierung eines Projekts?
- Wie sieht die Kalkulation eines sicherheitskritischen IT-Projekts aus?
- Welche Komponenten sind für die Kalkulation relevant?
- Welche Dokumente sind bei einem sicherheitskritischen IT-Projekt relevant?

- Wie sind in einem sicherheitskritischen IT-Projekt der Kosten- und der Zeitrahmen definiert?
- Welche Sicherheitsmechanismen können in einem sicherheitskritischen IT-System Anwendung finden?
- Welche Risiken müssen für den Projekterfolg nachverfolgt werden?
- Wie können Risiken bewertet und gemanagt werden?
- Welche Restriktionen müssen im Ausschreibungsverfahren berücksichtigt werden?
- Wie wird das Ausschreibungsrecht eingehalten?
- Welche Änderungen werden gegenüber dem originären Projekt-Scope benötigt und wie werden diese in den Projekt-Scope integriert?
- Welche Teile eines Projekts können fest bepreist werden, ohne ein zu hohes Risiko einzugehen?
- Welche Erfolgsfaktoren gibt es in sicherheitskritischen IT-Projekten?
- Welche Misserfolgsfaktoren gibt es in sicherheitskritischen IT-Projekten?
- Welche Risiken existieren in sicherheitskritischen IT-Projekten?
- Welche Komponenten hat ein sicherheitskritisches IT-System?
- Welche politischen Stakeholder müssen im Projekt berücksichtigt werden, um den Projekterfolg nicht zu gefährden?
- Welche Rechtfertigung des Projekts existiert im öffentlichen Interesse, wie z. B. die Sicherstellung der Versorgung der Bürger?
- Wie kann mithilfe von PRINCE2 ein sicherheitskritisches IT-Projekt durchgeführt werden?
- Welche Pläne müssen erstellt werden für die Durchführung eines sicherheitskritischen IT-Projekts?
- Welche Anforderungen (funktional, nicht-funktional und technisch) müssen berücksichtigt werden, um ein sicherheitskritisches IT-System aufzubauen?
- Gibt es Anforderungen für die Einführung und den Betrieb eines sicherheitskritischen IT-Systems?
- Wie muss ein Konzept oder eine IT-Architektur für ein sicherheitskritisches IT-System aussehen?
- Wie können moderne und zukunftsorientierte Cloud-Technologien im sicherheitskritischen Umfeld genutzt werden, während die Sicherheitsanforderungen weiterhin gewahrt werden?

- Wie und wann müssen Mitarbeiter geschult werden, um ein Bewusstsein über Gefahren adäquat sicherzustellen?
- Wie wird die Nutzerakzeptanz eines neuen IT-Systems sichergestellt?
- Welche Daten werden in einem sicherheitskritischen IT-System benötigt?
- Wie werden die Verfügbarkeit und Unverfälschtheit der Daten sichergestellt?
- Wie kann die Hochverfügbarkeit eines sicherheitskritischen IT-Systems sichergestellt werden?
- Wie kann sichergestellt werden, dass ein sicherheitskritisches IT-System trotz Redundanzanforderungen effizient und angemessen ist?
- Welche Dokumentation ist zielführend, um die Betriebsbereitschaft eines sicherheitskritischen IT-Systems sicherzustellen?
- Wie kann mithilfe einer vertraglichen Vereinbarung (nach EVB-IT) die Betriebsbereitschaft eines sicherheitskritischen IT-Systems sichergestellt werden?
- Welche Fehlertoleranz muss ein sicherheitskritisches IT-System vorweisen?

Die dargestellte Liste der Kompetenzfragen ist bei dem Entwicklungsprozess der sicherheitskritischen IT-Projekt-Ontologie stets zu berücksichtigen und bei Bedarf zu erweitern oder zu verkürzen. Das nachträgliche Herausnehmen einer Kompetenzfrage kommt in Frage, wenn sich im Nachgang herausstellt, dass sich die Kompetenzfrage als nicht relevant herausstellt. In diesem Beitrag wurden jedoch keine Kompetenzfragen herausgenommen. Stattdessen hat sich im Rahmen der Ontologiekonstruktion jedoch herausgestellt, dass sich weitergehende Fragestellungen ergeben haben. So hat sich beispielsweise ergeben, dass weitergehende Fragen hinsichtlich des Vergabeverfahrens notwendig waren, welche eine Auswirkung auf die Umsetzung eines sicherheitskritischen IT-Projekts haben. Zwar existieren beispielsweise Kompetenzfragen wie „Welche Restriktionen müssen im Ausschreibungsverfahren berücksichtigt werden?“ und „Wie wird das Ausschreibungsrecht eingehalten?“. Aber durch eine nähere Betrachtung der genannten Kompetenzfragen ergeben sich weitergehende Fragestellungen, wie beispielsweise „Welches Vergaberecht kann für sicherheitskritische IT-Projekte angewendet werden?“. Dieses Beispiel zeigt exemplarisch die Notwendigkeit, weitergehende Fragestellungen im Rahmen der Ontologiekonstruktion zu formulieren.

3.2.3.2 Prüfung bestehender Ontologien für ihre Nutzung zur Konstruktion einer neuen IT-Projekt-Ontologie

In diesem Schritt wird geprüft, ob eine ähnliche bestehende Ontologie als Grundlage genutzt werden kann. Es kann in Betracht gezogen werden, eine komplette Ontologie wiederzuverwenden oder auch nur Teile einer Ontologie.

Für die Suche nach potenziell wiederverwendbaren Ontologien stehen verschiedene „Ontologiebibliotheken“ zur Verfügung. Beispielhaft zu erwähnen sind DAML und die Protégé Ontology Library. In diesen Ontologiebibliotheken konnte zwar keine Ontologie gefunden werden, die sich für die Entwicklung einer Ontologie für sicherheitskritische IT-Projekte wiederverwenden lässt. Für diesen Beitrag eignen sich jedoch zwei Ontologien zur Wiederverwendung, die nicht in den einschlägigen Ontologiebibliotheken zu finden sind.

Für die Entwicklung einer sicherheitskritischen IT-Projekt-Ontologie wird eine Projektmanagement-Domänen-Ontologie (kurz: PM-Domänen-Ontologie) zugrunde gelegt, die im Rahmen des BMBF-geförderten Verbundprojekts KI-LiveS (KI-Labor für verteilte und eingebettete Systeme) durch das Institut für Produktion und Industrielles Informationsmanagement (PIM) der Universität Duisburg-Essen entwickelt wurde. Die zu erstellende sicherheitskritische IT-Projekt-Ontologie wird als Teilbereich in die PM-Domänen-Ontologie integriert.

Die Verwendung der PM-Domänen-Ontologie als Grundlage für die Konstruktion der sicherheitskritischen IT-Projekt-Ontologie hat für diesen Beitrag folgende Vorteile:

- Grundlegende Klassen, Relationen und Attribute des Projektmanagements sind in der PM-Domänen-Ontologie bereits vorhanden und sehr allgemein gefasst, sodass eine konkrete Einordnung von Begriffen als Subkategorien im Sinne einer Taxonomie vorgenommen werden kann.
- Die PM-Domänen-Ontologie wurde bereits mithilfe des CBR-Tools jCORA erprobt.
- Eine benutzerspezifische Anpassung ist ohne vorherige Anpassungsmaßnahmen möglich.
- Im Rahmen des KI-LiveS-Projekts wurde mittels der PM-Domänen-Ontologie eine PRINCE2- und Risikomanagement-Ontologie mit dem Fokus auf sicherheitskritische IT-Projekte entwickelt.

Die nachfolgende Abbildung 4 zeigt den Umfang der für diesen Beitrag verwendeten PM-Domänen-Ontologie in Protégé.

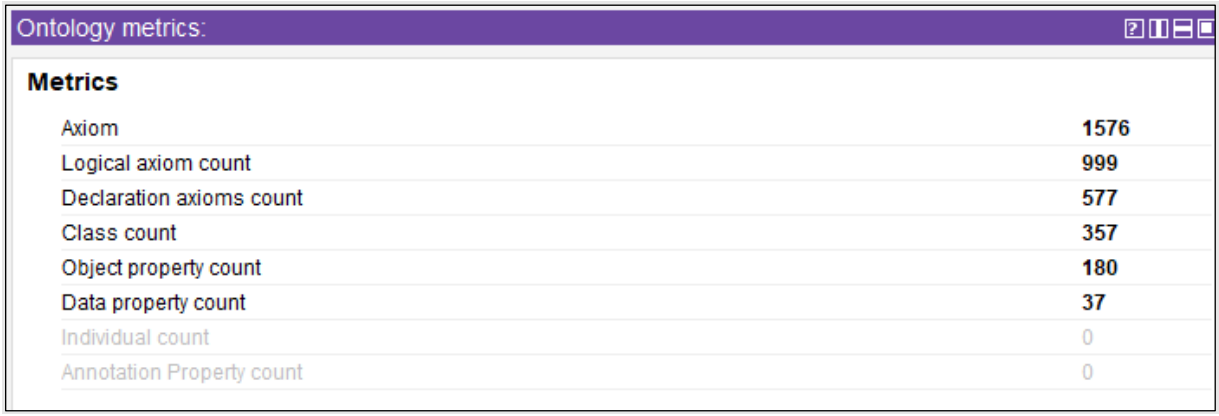


Abbildung 4: PM-Domänen-Ontologie

Jedoch lassen sich auch folgende Nachteile bei der Verwendung der PM-Domänen-Ontologie feststellen:

- Es liegt gegenwärtig keine aussagekräftige Dokumentation für die Komponenten der PM-Domänen-Ontologie vor.
- Die PM-Domänen-Ontologie ist eine sehr umfangreiche Ontologie, welche mit einer „Upper Ontology“ vergleichbar ist. Die Komplexität der Ontologie erfordert eine nicht zu vernachlässigende Einarbeitungszeit, die gegenwärtig, wie bereits angemerkt, ohne aussagekräftige Dokumentation erfolgen muss.
- Kardinalitäten und semantische Regeln sind in der PM-Domänen-Ontologie nicht hinterlegt. Daher erweist sich als Kritikpunkt, dass die PM-Domänen-Ontologie bislang noch nicht ausreichend spezifiziert wurde.
- Die PM-Domänen-Ontologie unterliegt einer fortlaufenden Entwicklung, sodass sich Klassen, Relationen und Attribute jederzeit ändern können. Es fehlen Versionsinformationen zu der Ontologie, sodass eine Versionierung schwer möglich ist.

Insgesamt betrachtet, überwiegen jedoch die zuvor genannten Vorteile die hier genannten Nachteile. Die Nachteile werden bewusst akzeptiert, um die PM-Domänen-Ontologie und die dazugehörige OWL-Datei um die Ontologiekomponenten der sicherheitskritischen IT-Projekte in Protégé zu erweitern.

Neben der PM-Domänen-Ontologie wird die PRINCE2- und Risikomanagement-Ontologie verwendet, die ebenfalls im Rahmen des KI-LiveS-Projekts entwickelt wurde. Die PRINCE2- und Risikomanagement-Ontologie wurde auf Basis der PM-Domänen-Ontologie erstellt und ist speziell für sicherheitskritische IT-Projekte ausgelegt. Die Wiederverwendung der PRINCE2- und Risikomanagement-Ontologie hat, um eine Ontologie für sicherheitskritische IT-Projekt zu konstruieren, folgende Vorteile:

- Die Ontologie ist für sicherheitskritische IT-Projekte entwickelt worden.
- Die Ontologie nutzt als Grundlage die PM-Domänen-Ontologie.
- Mithilfe der Ontologie wurden Beispielfälle für sicherheitskritische IT-Projekte im CBR-Tool jCORA erstellt.
- Die Verwendung von globalen Instanzen ermöglicht eine Zeitersparnis bei der Modellierung von IT-Projekten.
- Die PRINCE2- und Risikomanagement-Ontologie ist in der Publikation WEBER et al. (2021), S. 15-23 u. 50-75 ausführlich dokumentiert.

Die nachfolgende Abbildung 5 zeigt den Umfang der für diesen Beitrag verwendeten PRINCE2- und Risikomanagement-Ontologie für sicherheitskritische IT-Projekte.

Abbildung 5: Umfang der PRINCE2- und Risikomanagement-Ontologie

Die PRINCE2- und Risikomanagement-Ontologie unterliegt jedoch folgenden Einschränkungen, die bei der Konstruktion einer sicherheitskritischen IT-Projekt-Ontologie berücksichtigt werden müssen:

- Nicht alle Begriffe aus der Ontologie entstammen direkt dem PRINCE2-Standard.
- Nicht alle Bereiche und Begriffe des PRINCE2-Standards sind in der Ontologie ausreichend berücksichtigt.
- Kommentierungen durch Fachexperten, beispielsweise in Form von definierten Kompetenzfragen oder Begriffen, werden vermisst.
- Eine Integration der PRINCE2- und Risikomanagement-Ontologie in die zugrunde liegende PM-Domänen-Ontologie kann zu Widersprüchen führen, weil bei der Konstruktion der PRINCE2- und Risikomanagement-Ontologie einige Klassen aus der zugrunde liegenden PM-Domänen-Ontologie entfernt wurden, die für die Ausdrucksfähigkeit in Bezug auf sicherheitskritische IT-Projekte von Bedeutung sind.

Die Zielstellung bei der Konstruktion der PRINCE2- und Risikomanagement-Ontologie war es, die Ontologie für sicherheitskritische IT-Projekte auszulegen. Aufgrund dieses Hintergrunds wurden diverse Grundlagen wie die Integrationsmöglichkeiten in das CBR-Tool jCORA unter Verwendung von Fallbeispielen aus dem sicherheitskritischen IT-Projekt-Umfeld sowie die Berücksichtigung einiger (wenn auch nicht aller) sicherheitskritischer IT-Projektmerkmale in dieser PRINCE2- und Risikomanagement-Ontologie bereits berücksichtigt. Obwohl es einige Einschränkungen gibt, wie die aufgeführten Nachteile zeigen, werden diese Einschränkungen in der Ontologiekonstruktion in diesem Beitrag korrigiert.

3.2.3.3 Bestimmung wichtiger Begriffe für die IT-Projekt-Ontologie

Es wird eine möglichst vollständige Liste von Begriffen entwickelt, die für die sicherheitskritische IT-Projekt-Ontologie relevant sind. Für die praxisnahe Konstruktion einer Ontologie für sicherheitskritische IT-Projekte wurden Experten nach relevanten Begriffen aus der Domäne

von sicherheitskritischen IT-Projekten befragt und Begriffe aus einschlägigen Leistungsbeschreibungen von öffentlichen Ausschreibungen extrahiert, die auf sicherheitskritische IT-Projekte abzielen.

Bei der Aufzählung der relevanten Begriffe ist es aus der Sichtweise einer Ontologie unerheblich, ob es sich um potenzielle Klassen, Attribute oder Relationen handelt. Die Begriffssammlung dient als Auflistung von relevanten Begriffen, die im Konstruktionsprozess der Ontologie genutzt werden sollen. Die Aufzählung beinhaltet in diesem Schritt noch keine Konkretisierung zur späteren Verwendung der Begriffe. Vielmehr ist es das Ziel, eine umfassende Auflistung über die relevanten Begriffe zu erhalten, die es in der zu erstellenden Ontologie zu berücksichtigen gilt. Die nachfolgende Tabelle 3 stellt einen exemplarischen Auszug der identifizierten Begriffe für sicherheitskritische IT-Projekte dar.

Sicherheit	Vertraulichkeit	Datenschutz
Ausfallsicherheit	KRITIS	Gesetzeslage
Vergabeverfahren	Ausschreibung	Verfügbarkeit
Redundanzen	Sicherheitsstufe	Ausschreibungsrecht
Sicherheitskritisches IT-System	Total Contract Value (TCV)	Lizenzen
Hardware	Software	Systemintegration
OnPremise	Nutzer	Nutzen
Softwarequalität	EVB-IT-Vertrag	Vergabevolumen
Vergabebegleitung	Generalunternehmer	Subunternehmer
Softwareentwicklung	Programmiersprachen	Abhängigkeit
funktionale Anforderung	nicht-funktionale Anforderung	technische Anforderung
Betrieb	Datenmigration	Schulung
Angebotspräsentation	Rechenzentrum	Kundendaten
Ausschreibungsunterlagen	Systemumgebungen	Bid-Team
Erfolgsfaktoren	Misserfolgsfaktoren	Bundesland
Redundanzen	Sicherheitskonzept	Strategie
Akzeptanz	Behörden und Organisationen mit Sicherheitsaufgaben	Vergaberechtsverletzung
Einsatzleitsystem	Zugriffszahl	Schnittstellen
Gesamtabnahme	Teilabnahme	IT-Sizing
IT-Beratung	BSI-Grundschutz	Datenpflege
externer Projektaudit	Kernaufgaben	Service
Wartung	Robustheit	Rollen- und Rechtekonzept

Tabelle 3: Begriffe zu sicherheitskritischen IT-Projekten (exemplarischer Auszug)

Auf eine vollständige Auflistung der Begriffe aus dem PRINCE2-Standard wird an dieser Stelle verzichtet. Dennoch erfolgt eine exemplarische Auflistung von Begriffen des Risikomanagements, die u. a. in den bereits erwähnten Experteninterviews eine größere Rolle spielten.

Risikobeurteilung	Risikobewertung	Risikoidentifizierung
mittelbare Risiken	unmittelbare Risiken	Eintrittswahrscheinlichkeit
gesellschaftliche Auswirkung	Gefährdungslage	Meldepflicht BSI
technologische Risiken	Systemausfall	Reaktionszeiten
Wiederherstellungszeiten	Risiko Politik	Risiko Gesellschaft
Service Level Agreements	Hackerangriff	Risiko Wirtschaft
Haftungsrisiko	Haftungsbeschränkung	Imagegefährdung
Risikoregister		

Tabelle 4: Begriffe zum Risikomanagement (exemplarischer Auszug)

Die in den Tabelle 3 und 4 auszugsweise dargestellten Begriffe sind wesentliche sprachliche Ausdrucksmittel, die sowohl von den befragten Experten als auch in den einschlägigen Leistungsbeschreibungen als erforderlich erachtet wurden, um den Anwendungsbereich „sicherheitskritisches IT-Projekt“ sprachlich zu konzeptualisieren.

3.2.3.4 Klassenkonstruktion

Zum besseren Verständnis der später exemplarisch vorgestellten Klassen der sicherheitskritischen IT-Projekt-Ontologie erfolgen zunächst einige Vorbemerkungen zur Vorgehensweise bei der Klassenkonstruktion. Es werden Vorbemerkungen für folgende Punkte vorgenommen:

- Erläuterung der Klassenkonstruktion
- Relevanz der Begriffe für die Klassenkonstruktion
- Auswahl der Methode für die Klassenkonstruktion
- Kapitelstruktur für die Erläuterung der konstruierten Klassen
- Tabellenstruktur für die Erläuterung der konstruierten Klassen

Die Erläuterung der Klassenkonstruktion erfolgt, soweit es möglich ist, aufbauend auf den in diesem Beitrag konstruierten Klassen. Eine Erläuterung der Klasse aus der zugrunde gelegten PM-Domänen-Ontologie und der PRINCE2- und Risikomanagement-Ontologie wird nicht für alle Klassen durchgeführt. Es werden nur die Klassen erläutert, die zusätzlich im Rahmen dieses Beitrags konstruiert wurden. Jedoch müssen in einzelnen Fällen Klassen aus der PM-Domänen-Ontologie erläutert werden, wenn eine Erweiterung im taxonomischen Sinne erfolgt, um eine Einordnung in die Klasse zu begründen. Die Erläuterung umfasst, wie die inhaltliche Ausgestaltung der bestehenden Klasse aus der PM-Domänen-Ontologie erfolgt ist, um die Konstruktionsentscheidung zu begründen. Grundsätzlich wurden die erste und zweite Hierarchieebene

der PM-Domänen-Ontologie nicht verändert, weil keine Änderungen für diese Hierarchieebenen für notwendig erachtet wurden.

Die in Kapitel 3.2.3.3 genannte Begriffssammlung dient als Grundlage für die Klassenkonstruktion. Dazu müssen die Begriffe aus der Begriffssammlung identifiziert werden, die in der Ontologie durch Klassen repräsentiert werden sollen. Begriffe, die einen hohen Detaillierungsgrad besitzen, sollten als Instanzen betrachtet werden. Allgemeiner formulierte Begriffe sollten als Klassen definiert werden. In der praktischen Ontologiekonstruktion zeigt sich jedoch, dass die vorgenommene Unterscheidung letztlich subjektiv ist und daher – abhängig von den Entscheidungen des Ontologiekonstrukteurs – unterschiedlich begründet werden kann. Die Entscheidung, ob ein Begriff als Klasse oder als Instanz modelliert wird, erfolgt jeweils einzelfallbezogen.

Für die Konstruktion der Klassen werden zwar die Begriffe aus Kapitel 3.2.3.3 als Grundlage verwendet, aber es werden auch Klassen in diesem Beitrag konstruiert, die nicht zuvor als Begriffe genannt worden sind. Die Begriffe dienen als Orientierung und Konstruktionshilfe, stellen aber nicht das ausschließliche Mittel für die Klassenkonstruktion dar. Wie bereits beschrieben, stellen die als Grundlage genommenen Leistungsbeschreibungen neben den Begriffen, die von den Experten genannt wurden, einen wesentlichen Beitrag für die Konstruktion der Klassen dar.

In der Literatur existieren unterschiedliche Methoden für die Konstruktion von Klassen. Häufig genannt werden die folgenden drei Methoden (vgl. NOY/MCGUINNESS (2001), S. 6-7):

- Top-Down-Ansatz: Zuerst werden die allgemeinsten Klassen einer Domäne festgelegt. Danach werden die weiteren Unterklassen konstruiert, bis die unterste Ebene der Klassenhierarchie erreicht wird.
- Bottom-Up-Ansatz: Es wird mit der Festlegung der konkretesten Klasse begonnen. Die konkreteren Klassen auf den unteren Hierarchieebenen werden so lange zusammengefasst, bis die allgemeinste Klasse erreicht wird.
- Middle-Out-Ansatz: Der Top-Down- und der Bottom-Up-Ansatz werden miteinander kombiniert.

Für die Konstruktion der sicherheitskritischen IT-Projekt-Ontologie wird die Kombination des Top-Down- und Bottom-Up-Ansatzes genutzt. Dies wird damit begründet, dass die zugrunde gelegte PM-Domänen-Ontologie und die PRINCE2- und Risikomanagement-Ontologie bereits eine Klassenstruktur vorgeben und ein kombinierter Ansatz als zielführender angesehen wird. Eine ausschließliche Top-Down- oder Bottom-Up-Konstruktion war bei der praktischen Konstruktion der sicherheitskritischen IT-Projekt-Ontologie nicht immer möglich. Im Verlauf der Ontologiekonstruktion ergaben sich weitere Erkenntnisse, wie z. B. bei der späteren Fallerstellung, die eine Anpassung der sicherheitskritischen IT-Projekt-Ontologie erforderlich machten. In der praktischen Umsetzung hat es sich als praktikabel erwiesen, zunächst die wichtigsten

Klassen zu konstruieren und anschließend speziellere Unterklassen sowie allgemeinere Oberklassen.

Für eine verständliche Ex-post-Erläuterung der sicherheitskritischen IT-Projekt-Ontologie erscheint es jedoch angebracht, den Top-Down-Ansatz zu wählen, um die Konstruktion der Klassen leichter nachvollziehen zu können. Die Erläuterung der konstruierten Klassen erfolgt tabellarisch. Die jeweiligen Tabellen sind wie folgt aufgebaut:

Klasse	**Beschreibung**	**Subklasse von**

Tabelle 5: Tabellenstruktur für die Beschreibung der Klassen

In der Spalte „Beschreibung“ erfolgt die Beschreibung der konstruierten oder wiederverwendeten Klasse und in der Spalte „Subklasse von“ erfolgt die Benennung der übergeordneten Klasse.

Im Folgenden wird nur eine Auswahl von Klassen beschrieben, die nach Einschätzung der Autoren insbesondere für Leser ohne umfassende Erfahrungen mit Ontologien einen gut verständlichen Überblick über die Klassenstruktur der hier vorgestellten sicherheitskritischen IT-Projekt-Ontologie bieten. Leser, die an mehr Details dieser Ontologie interessiert sind, werden auf die umfassende Dokumentation der Klassenstruktur in SETHUPATHY (2024), S. 102-234 u. 645-677, verwiesen.

Die Klasse *Thing* ist die Maximalklasse oder „oberste“ Klasse der sicherheitskritischen IT-Projekt-Ontologie. Alle nachfolgenden Klassen sind Subklassen der Maximalklasse *Thing*. Die Klasse *Thing* stellt den Ausgangspunkt der Taxonomie dar und bildet die „nullte“ Hierarchieebene. Durch die Klasse *Thing* ist festgelegt, dass es in der Ontologie nur eine Taxonomie gibt. Die Existenz von mehreren, parallelen Taxonomien wird durch die Verwendung der Klasse *Thing* ausgeschlossen.

Als erste Subklasse der Klasse *Thing* folgen die Klassen *Eigenschaft* und *Objekt*. Beide Subklassen entstammen der PM-Domänen-Ontologie. Die nachfolgende Abbildung 6 zeigt die Maximalklasse *Thing* und die direkt untergeordneten Subklassen *Eigenschaft* und *Objekt* auf der ersten Hierarchieebene.

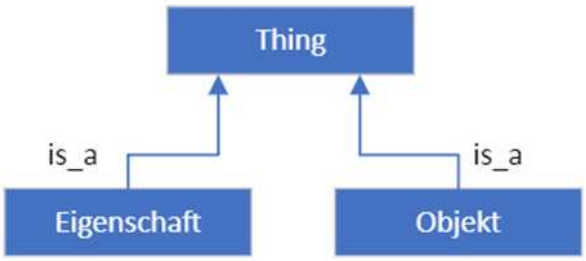


Abbildung 6: Subklassen der Klasse *Thing*

Zunächst erfolgt die Erläuterung der Klasse *Eigenschaft* exemplarisch ausgewählten, zugehörigen Subklassen. Später wird die Klasse *Objekt* mit exemplarisch ausgewählten, zugehörigen Subklassen beschrieben.

Die Klasse *Eigenschaft* ist in der ersten Hierarchieebene durch die verwendete PM-Domänen-Ontologie vorgegeben und wird in der sicherheitskritischen IT-Projekt-Ontologie wie folgt charakterisiert:

- Die Klasse *Eigenschaft* ist von den Ontologiekomponenten der Attribute und Relationen, die auch als Merkmale einer Klasse bezeichnet werden, zu unterscheiden. Bei der generellen Konstruktion der Klassen für die sicherheitskritische IT-Projekt-Ontologie zeigt sich, dass eine Abgrenzung zwischen Subklassen der Klasse Eigenschaft und den Merkmalen (in Form von Attributen und Relationen) einer Klasse nicht trivial ist. Im Einzelfall kann eine Abgrenzung für die Unterscheidung fließend sein und unterliegt der Subjektivität des Ontologiekonstrukteurs. Exemplarisch wird diese fließende Unterscheidung durch die Subklassen der Klasse *InformationstechnischeEigenschaft* veranschaulicht. Diese wurden als Subsubklassen der Klasse *Eigenschaft* konzipiert, könnten jedoch auch als Eigenschaften durch die Verwendung primitiver Datentypen innerhalb einer Klasse wie etwa der Klasse *Hardware* konstruiert worden sein. Dieses Beispiel soll an dieser Stelle die Herausforderung der Unterscheidung illustrieren und betont die subjektive Natur einer Ontologie sowie deren Konstruktion (Design“) als einen kreativen, nicht vollständig „objektivierbaren“ Prozess.
- Die Subklassen der Klasse *Eigenschaft* stellen sprachliche Ausdrucksmittel für die Eigenschaftsbeschreibungen für die Subklassen der Klasse *Objekt* zur Verfügung, welche mittels nicht-taxonomischer Relationen verbunden werden und eine besondere Bedeutung für die sicherheitskritische IT-Projekt-Ontologie besitzen. Erwähnenswerte Eigenschaften sind beispielsweise informationstechnische Eigenschaften, Umgebungsbedingungen und Vergabeverfahrensarten. Diese Eigenschaften dienen dazu, spezielle Eigenschaften zu modellieren, die mit einem Attribut oder einer Relation nur unzureichend ausgedrückt werden können.
- Eigenschaften zeichnen sich durch ihre Unveränderlichkeit aus, Objekte hingegen können sich verändern.

Im Folgenden wird zunächst auf Subklassen der Klasse *Eigenschaft* eingegangen.

Die Klasse *Eigenschaft* ist in der PM-Domänen-Ontologie in die drei Subklassen *InstanzBeschreibungskonstituente*, *QualitativeEigenschaft* und *QuantitativeEigenschaft* unterteilt, wie in der nachfolgenden Abbildung 7 dargestellt ist. Die Subklassen stellen die zweite Hierarchieebene der sicherheitskritischen IT-Projekt-Ontologie dar.

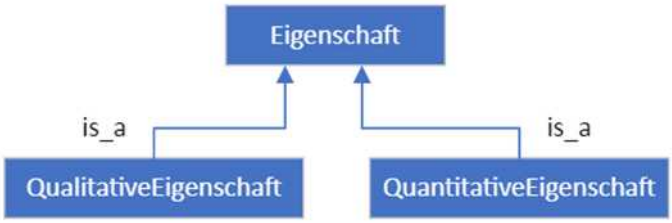


Abbildung 7: Zweite Hierarchieebene der Klasse *Eigenschaft*

Das Unterscheidungskriterium zwischen den drei Subklassen ist, ob der Eigenschaft eine natürlich sprachlich verfasste Merkmalskomponente zugrunde liegt oder ob diese eine messbare Kennzahl als Merkmalskomponente hat. Die Klasse Eigenschaft und die zugehörigen Subklassen werden in der nachfolgenden Tabelle erläutert.

Klasse	**Beschreibung**	**Subklasse von**
QualitativeEigenschaft	Die Klasse dient als Oberklasse für alle qualitativen Eigenschaften. Qualitative Eigenschaften sind alle nicht-numerischen, natürlich sprachlich verfassten Eigenschaften.	Eigenschaft
QuantitativeEigenschaft	Die Klasse dient als Oberklasse für alle quantitative Eigenschaften. Quantitative Eigenschaften sind alle numerischen, mittels einer messbare Kennzahl Eigenschaften.	Eigenschaft

Tabelle 6: Beschreibung der Subklassen der Klasse *Eigenschaft*

Die Klasse *QualitativeEigenschaft* enthält verschiedene Subklassen, wie die nachfolgende Abbildung 8 darstellt.

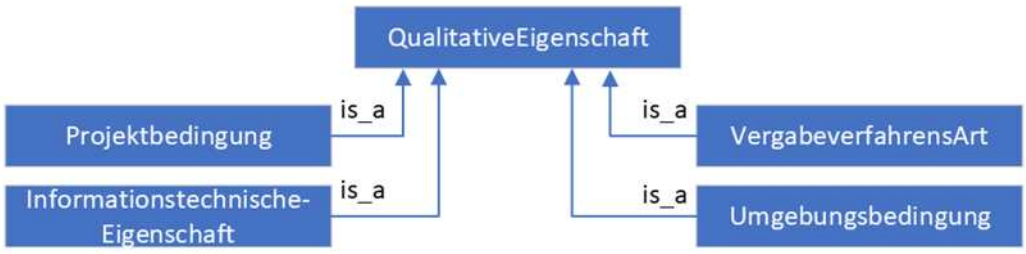


Abbildung 8: Subklassen der Klasse *QualitativeEigenschaft*

In dieser dritten Hierarchieebene wurden die Klassen *InformationstechnischeEigenschaft*, *Umgebungsbedingung*, *Projektbedingung* und *VerfahrensArt* als Subklassen der Klasse *QualitativeEigenschaft* hinzugefügt. Auf die genannten Klassen wird in der nachfolgenden Tabelle 7 näher eingegangen. Sie dienen hauptsächlich zur Strukturierung von allgemeingültigen qualitativen Eigenschaften von Projektbedingungen, öffentlichen Vergabearten und Umgebungsbedingungen mit dem Fokus auf sicherheitskritische IT-Projekte.

Klasse	**Beschreibung**	**Subklasse von**
Projektbedingung	Die Klasse Projektbedingung stellt in diesem Kontext eine Voraussetzung für ein Projekt dar, die als unternehmensinterne Bedingung dient und im Gegensatz zur Umgebungsbedingung keinen Einfluss von außerhalb des Unternehmens wiedergibt.	QualitativeEigenschaft
Informationstechnische Eigenschaft	Die Klasse subsumiert verschiedene Informationstechnische Eigenschaften, die für sicherheitskritische IT-Projekte relevant sind.	QualitativeEigenschaft
Umgebungsbedingung	Die Klasse subsumiert verschiedene Umgebungsbedingungen, die auf ein sicherheitskritisches IT-Projekt als externe Faktoren einwirken. Diese Umgebungsbedingungen dienen dazu, die „weichen" Faktoren eines Projekts zu berücksichtigen. Eine Umgebungsbedingung stellt in diesem Zusammenhang einen Zustand unternehmensexterner Einflüsse dar.	QualitativeEigenschaft
VergabeverfahrensArt	In der öffentlichen Vergabe existieren in Deutschland verschiedene Arten der Vergabe. Die aktuellen Vergabeverfahrensarten werden mittels dieser Klasse beschrieben.	QualitativeEigenschaft

Tabelle 7: Beschreibung der Subklassen der Klasse *QualitativeEigenschaft*

Die Klasse *QuantitativeEigenschaft* umfasst die Klassen DerivateQuantitativeEigenschaft und *OrginäreQuantitativeEigenschaft*. Beide Klassen sind bereits durch die PM-Domänen-Ontologie vorgegeben. Die nachfolgende Abbildung 9 zeigt die Klasse *QuantitativeEigenschaft* mit den genannten Subklassen.

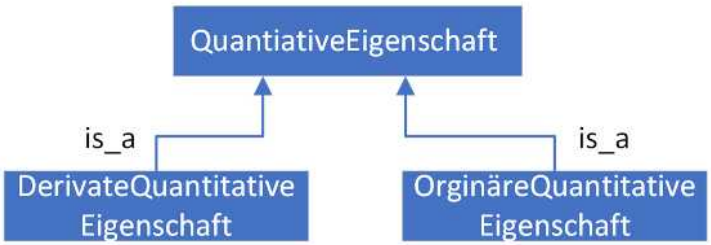


Abbildung 9: Subklassen der Klasse *QuantitativeEigenschaft*

Die beiden Subklassen der Klasse *QuantitativeEigenschaft* werden in der sicherheitskritischen IT-Projekt-Ontologie verwendet, um zwischen abgeleiteten Kennzahlen und originären Kennzahlen zu unterscheiden. Da sicherheitskritische IT-Projekte durch eine öffentliche Vergabe an einen Bieter vergeben werden, sollen sich beispielsweise die abgeleiteten Kennzahlen aus Eignungskriterien, Ausschlusskriterien und Zuschlagskriterien auch in der Klassenkonstruktion wieder finden. Auch Mengengerüste als abgeleitete Kennzahlen spielen für sicherheitskritische IT-Projekte eine wichtige Rolle. Originäre quantitative Eigenschaften hingegen stellen einfache Kennzahlen dar. Zum Beispiel werden Mengenangaben für die Dimensionierung von IT-Systemen verwendet. Die Kennzahlen werden als faktisch gegeben angesehen und unterscheiden sich von abgeleiteten quantitativen Eigenschaften dadurch, dass sie nicht für eine abgeleitete Bewertung verwendet werden.

Klasse	**Beschreibung**	**Subklasse von**
Derivative Quantitative Eigenschaft	Die Klasse *DerivativeQuantitative Eigenschaft* subsumiert messbare Eigenschaften, die mittels Kennzahlen dargestellt werden, wobei der primäre Zweck der Kennzahlen die Ableitung einer Bewertung darstellt (z. B. Eignungskriterien, Ausschlusskriterien, Zuschlagskriterien).	QuantitativeEigenschaft
Orginäre Quantiative Eigenschaft	Die Klasse *OriginäreQuantitative Eigenschaft* umfasst messbare Eigenschaften, die mittels Kennzahlen dargestellt werden, ohne dass die Absicht besteht, daraus eine Bewertung abzuleiten.	QuantitativeEigenschaft

Tabelle 8: Beschreibung der Subklassen der Klasse *QuantitativeEigenschaft*

Die Klasse *InformationstechnischeEigenschaft* beschreibt die relevanten Eigenschaften eines sicherheitskritischen IT-Systems. Die nachfolgende Abbildung 10 stellt diese Klasse mit den zugehörigen Subklassen dar.

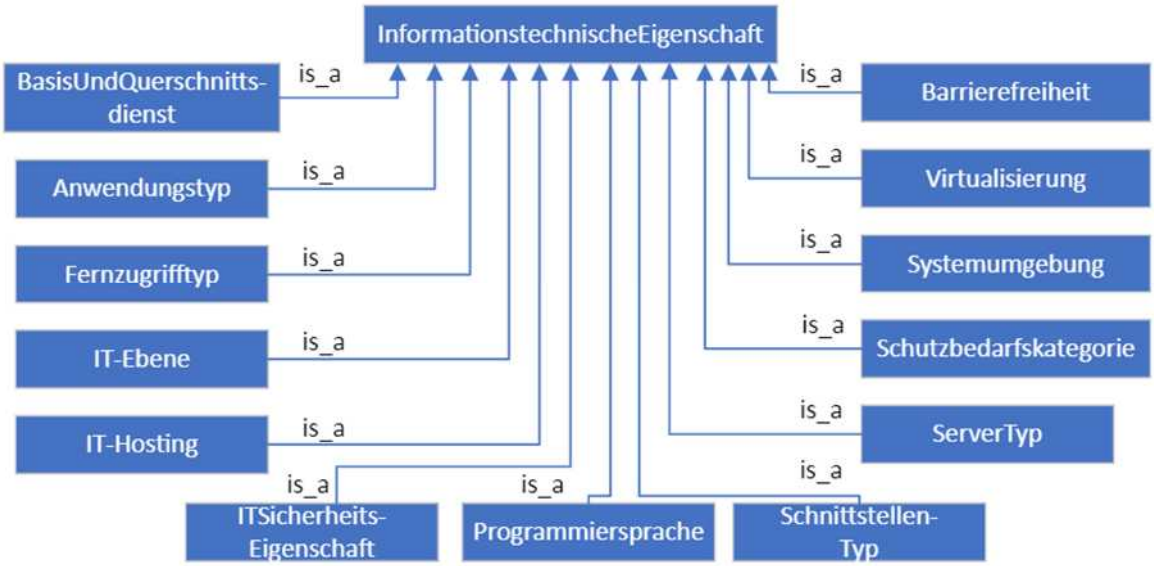


Abbildung 10: Subklassen der Klasse *InformationstechnischeEigenschaft*

Die informationstechnischen Eigenschaften dienen dazu, sicherheitskritische IT-Systeme hinsichtlich ihrer möglichen technischen Eigenschaften zu repräsentieren. Die nachfolgende Tabelle 9 beschreibt die jeweiligen Subklassen der informationstechnischen Eigenschaften.

Klasse	**Beschreibung**	**Subklasse von**
BasisUndQuerschnitts-dienste	Die Klasse *BasisUndQuerschnitts dienste* dient als Oberklasse aller Basis- und Querschnittsdienste in einem sicherheitskritischen IT-System. Dienste, die eine gemeinsame, querschnittliche Grundlage für andere, darauf aufbauende Dienste, z. B. Fachdienste, bilden, sind keiner einzelnen fachlichen Aufgabe direkt zugeordnet.	Informationstechnische Eigenschaft
Anwendungstyp	Die Klasse umfasst verschiedene Anwendungstypen, die aktuell in der Informationstechnologie verfügbar sind.	Informationstechnische Eigenschaft
Fernzugrifftyp	Die Klasse stellt die Oberklasse für die verschiedenen Fernzugriffstypen auf IT-Systeme dar. Unter Fernzugriff wird der externe Zugriff (außerhalb	Informationstechnische Eigenschaft

	des eigenen Netzwerks) auf das sicherheitskritische IT-System verstanden.	
IT-Ebene	Die Klasse stellt die Oberklasse für die IT-Ebenen eines IT-Systems dar.	Informationstechnische Eigenschaft
IT-Hosting	Die Klasse umfasst alle Arten des IT-Hostings für sicherheitskritische IT-Systeme.	Informationstechnische Eigenschaft
ITSicherheits Eigenschaft	Die Klasse beinhaltet alle Arten von IT-Sicherheitseigenschaften eines sicherheitskritischen IT-Systems. Die IT-Sicherheitseigenschaften stellen elementare Eigenschaften eines sicherheitskritischen IT-Systems dar, um die Schutzziele der KRITIS-Anforderung zu gewährleisten. IT-Sicherheitseigenschaften sind gekennzeichnet durch die Kombination mehrerer einzelner IT-Sicherheitseigenschaften.	Informationstechnische Eigenschaft
Programmiersprache	Die Klasse repräsentiert alle Arten von Programmiersprachen, die in sicherheitskritischen IT-Projekten verwendet werden können.	Informationstechnische Eigenschaft
Schnittstellen Typ	Die Klasse subsumiert alle IT-Schnittstellentypen, die in sicherheitskritischen IT-Projekten zum Einsatz kommen könnten. Die Schnittstellentypen umfassen offene und nicht-offene Standards. Bei IT-Schnittstellentypen handelt es sich um datenorientierte Übergänge zwischen Programmen oder Diensten, über die ein Datenaustausch stattfindet.	Informationstechnische Eigenschaft
ServerTyp	Die Klasse *ServerTyp* subsumiert verschiedene Servertypen, die aus Applikationssicht für die Implementierung eines sicherheitskritischen IT-Systems genutzt werden können. Die Klasse umfasst nicht den Servertyp aus Hardwaresicht.	Informationstechnische Eigenschaft

Schutzbedarfskategorie	Die Klasse dient als Oberklasse für die Schutzbedarfskategorien. Der Schutzbedarf dient zur Einstufung eines sicherheitskritischen IT-Systems.	Informationstechnische Eigenschaft
Systemumgebung	Die Klasse repräsentiert alle Arten von Systemumgebungen, die für sicherheitskritische IT-Systeme verwendet werden.	Informationstechnische Eigenschaft
Virtualisierung	Die Klasse dient als Oberklasse für alle Arten von Virtualisierung von IT-Hardware-Komponenten.	Informationstechnische Eigenschaft
Barrierefreiheit	Die Klasse dient als Oberklasse für die Eigenschaften der Barrierefreiheit einer informationstechnischen Anwendung.	Informationstechnische Eigenschaft

Tabelle 9: Beschreibung der Subklassen der Klasse *InformationstechnischeEigenschaft*

Die Klasse *VergabeverfahrensArt* repräsentiert mit ihren Subklassen alle Arten von öffentlichen Vergabeverfahren. Die nachfolgende Abbildung 11 zeigt die Klasse *VergabeverfahrensArt* mit ihren Subklassen.

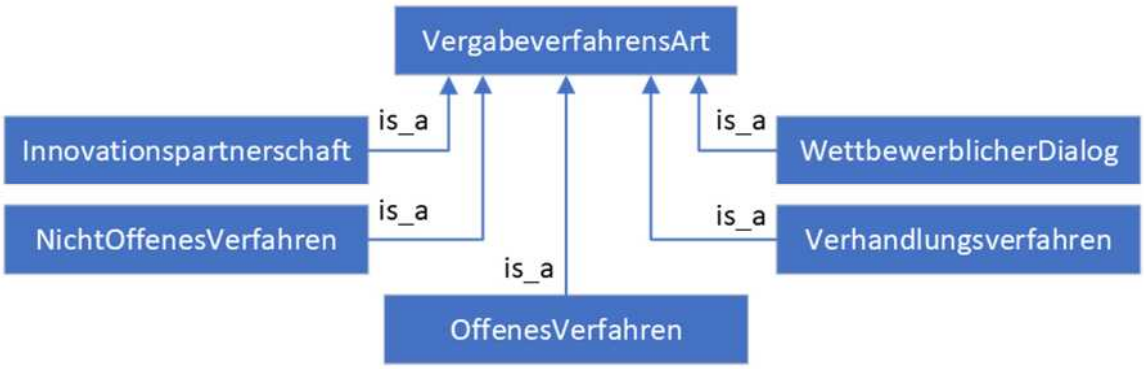


Abbildung 11: Subklassen der Klasse *VergabeverfahrensArt*

Öffentliche Vergabeverfahren sind im hohen Maße der Offenheit, der Transparenz und dem Grundsatz der Fairness verpflichtet, da mit öffentlichen Geldern Beschaffungen im öffentlichen Interesse erfolgen. Eine fehlende Beschränkung und Transparenz können einem Vergabeverfahren den Anschein von willkürlicher Beschaffung verleihen und den Verdacht einer Beeinflussung nahelegen.

Ein Vergabeverfahren im Rahmen von sicherheitskritischen IT-Projekten stellt eine besondere Herausforderung an Transparenz und Rechtssicherheit dar. Die Vergabe eines sicherheitskritischen IT-Systems kann ein bis zwei Jahre dauern, bevor die eigentliche Vergabe der Leistung

erfolgt. Hierbei ist eine mögliche Klage eines unterlegenen Bieters, die zu einer weiteren Verzögerung führen kann, noch nicht berücksichtigt. Somit beeinflusst ein Vergabeverfahren auch das sicherheitskritische IT-Projekt.

Die nachfolgende Abbildung12 illustriert die für die jeweiligen Vergabeverfahrensarten vorgesehenen Vergabeschritte. In sicherheitskritischen IT-Projekten kann es vorkommen, dass die Vergabeunterlagen einer Geheimhaltungsstufe unterliegen, sodass alle relevanten Unterlagen aus einem Vergabeverfahren einer Geheimhaltungsklassifizierung unterliegen.

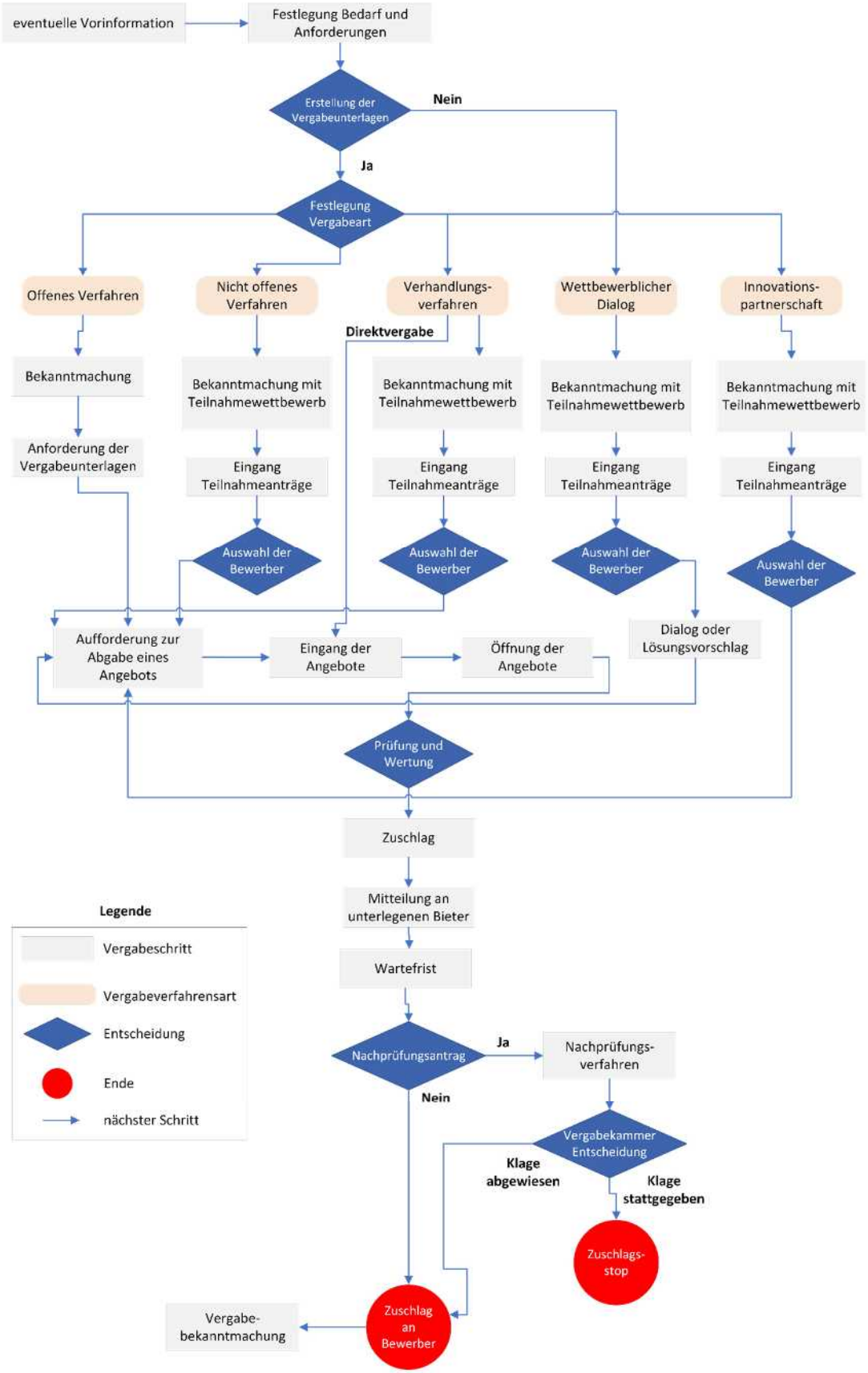


Abbildung 12: Vergabeverfahren

Die nachfolgende Tabelle 210 erläutert die Subklassen der Klasse *VergabeverfahrensArt*.

Klasse	**Beschreibung**	**Subklasse von**
Innovationspartnerschaft	Die Klasse *Innovationspartnerschaft* repräsentiert die Vergabeverfahrensart für eine Innovationspartnerschaft. Die Innovationspartnerschaft ist ein Verfahren zur Entwicklung innovativer, noch nicht auf dem Markt verfügbarer Produkte zum anschließenden Erwerb der daraus hervorgehenden Leistungen. Eine Innovationspartnerschaft ist eine Vergabeverfahrensart, die 2016 hinzugekommen ist.	VergabeverfahrensArt
NichtOffenesVerfahren	Die Klasse *NichtOffenes Verfahren* stellt die sprachlichen Ausdrucksmittel für das nicht offene Vergabeverfahren bereit. Das nicht offene Vergabeverfahren ist eine Vergabeverfahrensart, in der der öffentliche Auftraggeber nach vorheriger öffentlicher Aufforderung zur Teilnahme (Teilnahmewettbewerb) eine beschränkte Anzahl von Unternehmen stets gewährleistet nach objektiven, transparenten und nichtdiskriminierenden Kriterien auswählt, die er zur Abgabe von Angeboten auffordert.	VergabeverfahrensArt
OffenesVerfahren	Die Klasse *OffenesVerfahren* repräsentiert das offene Vergabeverfahren. Das offene Vergabeverfahren ist eine Vergabeverfahrensart, in der der öffentliche Auftraggeber eine unbeschränkte Anzahl von Unternehmen öffentlich zur Abgabe von Angeboten auffordert.	VergabeverfahrensArt

Verhandlungsverfahren	Die Klasse *Verhandlungsverfahren* repräsentiert das Verhandlungsverfahren. Das Verhandlungsverfahren ist eine Vergabeverfahrensart, bei der sich der öffentliche Auftraggeber mit oder ohne Teilnahmewettbewerb an ausgewählte Unternehmen wendet, um mit einem oder mehreren dieser Unternehmen über die Angebote zu verhandeln.	VergabeverfahrensArt
WettbewerblicherDialog	Die Klasse *Wettbewerblicher Dialog* bietet die sprachlichen Ausdrucksmittel zur Beschreibung der Vergabeverfahrensart Wettbewerblicher Dialog. Der wettbewerbliche Dialog ist ein Vergabeverfahren zur Vergabe öffentlicher Aufträge mit dem Ziel der Ermittlung und Festlegung der Mittel, mit denen die Bedürfnisse des öffentlichen Auftraggebers am besten erfüllt werden können. Der öffentliche Auftraggeber führt mit den ausgewählten Unternehmen einen Dialog zur Erörterung aller Aspekte der Auftragsvergabe.	VergabeverfahrensArt

Tabelle 2: Beschreibung der Subklassen der Klasse *VergabeverfahrensArt*

Die Umgebungsbedingungen beeinflussen ein sicherheitskritisches IT-Projekt auf verschiedene Art und Weise, was durch die nachfolgenden Subklassen ausgedrückt wird. Die nachfolgende Abbildung13 stellt die Klasse *Umgebungsbedingung* mit ihren Subklassen dar.

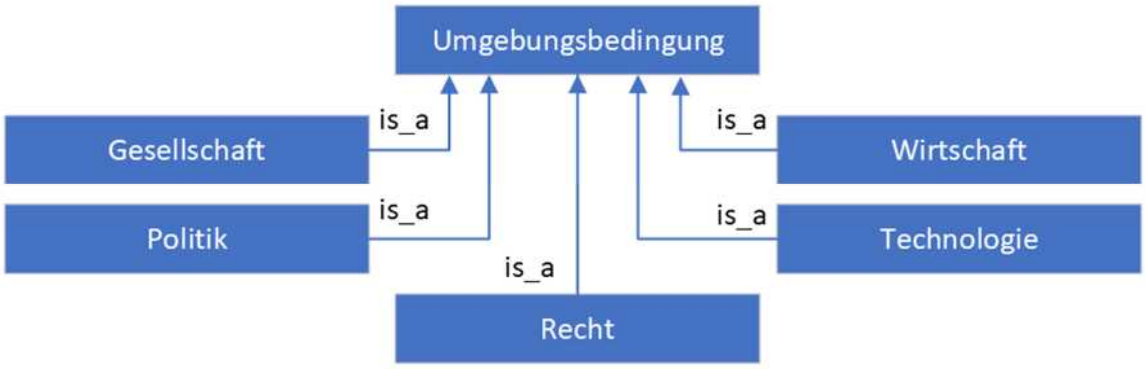


Abbildung 13: Subklassen der Klasse *Umgebungsbedingung*

Die Bedeutung von Umgebungsbedingungen ist teilweise als Rahmenbedingungen in Leistungsbeschreibungen von sicherheitskritischen IT-Projekten zu finden, damit der potenzielle Anbieter die Umgebungsbedingungen kennt. Die Unterteilung in die nachfolgenden Subklassen erfolgte nach Maßgabe des PEST-Konzepts. Der Begriff PEST setzt sich aus den vier englischen Termini „Political“, „Economic“, „Sociological“ und „Technological“ zusammen. Für die nachfolgende Betrachtung werden die vier Dimensionen des PEST-Konzepts um die Dimension „Recht” erweitert. Die Erweiterung begründet sich dahingehend, dass die sicherheitskritischen IT-Projekte im Bereich der öffentlichen Vergaben einzuordnen sind und somit das Vergaberecht eine wesentliche Rolle spielt. Die nachfolgende Tabelle 11 stellt die Oberklasse *Umgebungsbedingung* mit ihren Subklassen dar.

Klasse	**Beschreibung**	**Subklasse von**
Gesellschaft	Die Klasse *Gesellschaft* subsumiert alle Umgebungsbedingungen, die gesellschaftlich beeinflusst werden. Gesellschaftliche Umgebungsbedingungen können zum einen Strukturmerkmale (Bevölkerungsstruktur, Einkommensstruktur, Anteil der Stadtbewohner, Bildungswesen etc.) und zum anderen dazugehörige Trends (z. B. demographischer Wandel, gestiegenes Sicherheitsbedürfnis) sein. Auch Ansprüche an Produkte oder an Unternehmen selbst sind unter gesellschaftlichen Umgebungsbedingungen zusammengefasst (steigendes Umwelt- und Gesundheitsbewusstsein, Individualisierung, digitale Souveränität, Datenschutzsensibilität der Bürger).	Umgebungs bedingung
Politik	Die Klasse *Politik* subsumiert die Umgebungsbedingungen der Politik. Der Umgebungsfaktor „Politik“ umfasst den Einflussfaktor der Politik als oberster Verantwortlicher für die Beschaffung eines sicherheitskritischen IT-Systems. Die politischen Rahmenbedingungen hängen von der jeweiligen Legislaturperiode mit den dazugehörigen politischen Agenden ab und sind	Umgebungs bedingung

	unabhängig von einer Projektdauer. Der Faktor „Politik" umfasst in diesem Fall nicht die rechtlichen Rahmenbedingungen, diese werden im folgenden Faktor „Recht" ausgedrückt.	
Recht	Die Klasse *Recht* stellt die sprachlichen Ausdrucksmittel für die rechtlichen Umgebungsbedingungen zur Verfügung. Die rechtlichen Rahmenbedingungen sind auf kommunaler, regionaler, nationaler und supranationaler Ebene definiert. Sie umfassen die rechtlichen Rahmenbedingungen für den öffentlichen Auftraggeber und den Auftragnehmer. Darunter fallen Regelungen der Unternehmensverfassung, Vergaberecht, Verpflichtung nach VerpflG, Verpflichtungserklärung zur Einhaltung der ILO-Kernarbeitsnormen, Verpflichtung zur Behandlung von Verschlusssachen des Geheimhaltungsgrades VS-NfD, Datenschutzbestimmungen, Besteuerung, Haftungsbedingungen nach EVB-IT, No-Spy-Regelung inkl. technischer No-Spy-Klausel, Sicherheitsvorschriften nach BSI, KRITIS-Verordnung, Onlinezugangsgesetz sowie der Barrierefreiheit. Sicherheitskritische IT-Projekte stehen vor der Herausforderung, auf Bundes- und Landesebene bestehende rechtliche Vorgaben umsetzen zu müssen.	Umgebungs bedingung
Technologie	Die Klasse „Technologie" subsumiert die technologischen Umgebungsbedingungen. Technologische Entwicklungen beeinflussen IT-Projekte auf vielfältige Art und Weise. Technologie-Trends können dazu führen, dass neue Technologien entstehen oder auch bestehende Technologien abgekündigt werden. Der Aspekt der Technologie-Sicherheit spielt für sicherheitskritische IT-Projekte eine wichtige Rolle. Die eingesetzte Technologie in sicherheitskritischen IT-Projekten muss die höchsten Anforderungen an Betriebssicherheit, und Zukunftssicherheit der Technologie erfüllen. Die Handlungsfähigkeit der Behörden und Organisationen mit Sicherheitsaufgaben muss zu jedem Zeitpunkt gewährleistet sein. Größere Beratungskonzerne geben jedes Jahr eine Darstellung der jährlichen IT-Trends, die bei IT-Projekten bei der technologischen Implementierung berücksichtigt werden sollten, heraus.	Umgebungs bedingung
Wirtschaft	Die Klasse *Wirtschaft* ist die Oberklasse für die Umgebungsbedingungen der Wirtschaft. Wirtschaftliche Umgebungsbedingungen spielen bei öffentlichen Vergaben eine wichtige Rolle. Die ausschreibenden Stellen (Bund/Bundesländer/Kommunen) haben unterschiedliche wirtschaftliche Rahmenbedingungen. Auch die Bundesländer oder Kommunen selbst haben unterschiedliche Rahmenbedingungen auf der wirtschaftlichen Ebene.	Umgebungs bedingung

	So kann ein Bundesland wirtschaftlich besser aufgestellt sein als ein anderes Bundesland. Dies kann sich bei öffentlichen Vergaben widerspiegeln.	

Tabelle 11: Beschreibung der Subklassen der Klasse *Umgebungsbedingung*

Die nachfolgende Abbildung 14 stellt die Oberklasse *DerivativeQuantitativeEigenschaft* mit ihren Subklassen dar.

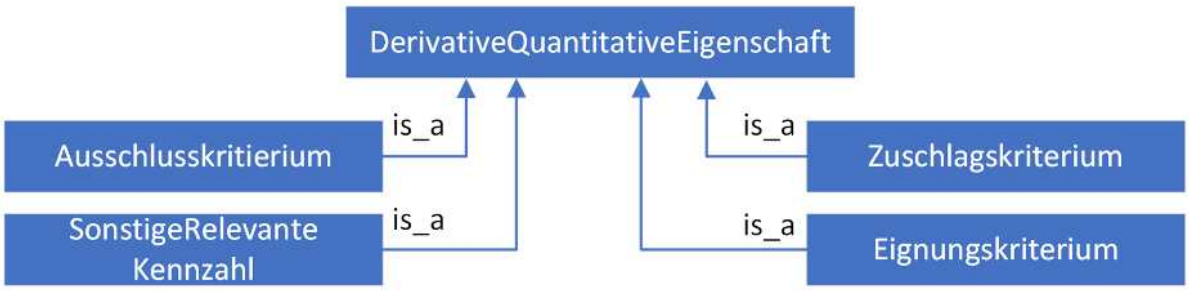


Abbildung 14: Subklassen der Klasse *DerivativeQuantitativeEigenschaft*

Die derivativen quantitativen Eigenschaften stellen Kennzahlen dar, die eine abgeleitete Bewertung ermöglichen. Die Klasse *DerivativeQuantitativeEigenschaft* lässt sich anhand der Eignungs-, Zuschlags- und Ausschlusskriterien aus einem Vergabeverfahren gut erklären. Anhand dieser Kriterien kann in einer öffentlichen Vergabe abgeleitet werden, ob ein Anbieter beispielsweise einen Teilnahmewettbewerb erfolgreich bestehen kann oder in einem Vergabeverfahren ausgeschlossen wird, da er die Eignungskriterien nicht erfüllt.

Im Folgenden wird auf die vier Subklassen *Ausschlusskriterium*, *Eignungskriterium SonstigeRelevanteKennzahl* und *Zuschlagskriterium* eingegangen.

Klasse	**Beschreibung**	**Subklasse von**
Ausschluss-kritierium	Die Klasse *Ausschlusskriterium* stellt die Oberklasse für Ausschlusskriterien in öffentlichen Vergaben dar. Allgemeingültige Ausschlusskriterien lassen dem öffentlichen Auftraggeber keinen Ermessensspielraum. In der Bekanntmachung eines Teilnahmeantrags oder in der Bekanntmachung einer Ausschreibung stehen neben allgemeingültigen Ausschlusskriterien darüber hinaus weitere inhaltliche Ausschlusskriterien, die bei der Angebotsbearbeitung durch einen Bieter berücksichtigt werden müssen.	DerivativeQuantitative Eigenschaft

Sonstige Relevante Kennzahl	Die Klasse *SonstigeRelevanteKennzahl* subsumiert als Oberklasse die sonstigen relevanten Kennzahlen.	DerivativeQuantitative Eigenschaft
Zuschlags-kriterium	Die Klasse *Zuschlagskriterium* repräsentiert die Oberklasse der Zuschlagskriterien im Rahmen eines Vergabeverfahrens. Unter den Zuschlagskriterien sind Auswahlkriterien des öffentlichen Auftraggebers zu verstehen, die dieser für seine Auftragsvergabe heranzieht. Die öffentlichen Auftraggeber haben die Pflicht, die Zuschlagskriterien, die auch diverse Unterkategorien besitzen können, in den Vergabeunterlagen der Ausschreibung offenzulegen. Die Zuschlagskriterien können eine unterschiedliche Gewichtung in der Bewertung besitzen. Auch die Gewichtung muss bekannt gemacht werden. Während eines Vergabeverfahrens die Zuschlagskriterien zu ändern, ist nicht zulässig. Die Zuschlagskriterien bringen zum Ausdruck, auf welche Eigenschaften der Auftraggeber im Wesentlichen Wert legt und was als Bewertungsgrundlage des Angebots dient. Für die potenziellen Auftragnehmer ist es von besonderer Bedeutung die Zuschlagskriterien zu erfüllen.	DerivativeQuantitative Eigenschaft
Eignungs-kriterium	Die Klasse *Eignungskriterium* dient als Oberklasse aller Eignungskriterien in einer öffentlichen Vergabe. Gemäß den GWB-Anforderungen gilt ein potenzieller Bieter als geeignet, wenn er die in den Vergabeunterlagen festgelegten Eignungskriterien erfüllt. Die Eignungskriterien sind von den Zuschlagskriterien zu unterscheiden. Die Eignungskriterien geben an, ob ein potenzieller Anbieter geeignet ist, ein Angebot zu unterbreiten. Diese Eignungskriterien dürfen ausschließlich Folgendes umfassen: • Erlaubnis und Befähigung zur Berufsausübung, • finanzielle und wirtschaftliche Leistungsfähigkeit, • berufliche und technische Leistungsfähigkeit. Die Eignungskriterien müssen in der Bekanntmachung der Vergabe in der Gesamtheit aufgezählt werden. Die Kriterien müssen verhältnismäßig sein und ausschließlich dazu dienen, die Unternehmen zu ermitteln, die zur gefragten Leistungserbringung geeignet sind.	DerivativeQuantitative Eigenschaft

	In der Regel werden Eignungskriterien in Teilnahmewettbewerben definiert, um aus Sicht des Auftraggebers geeignete Teilnehmer zur Aufforderung eines Angebots zu ermitteln.	

Tabelle 12: Beschreibung der Subklassen der Klasse *DerivativeQuantitativeEigenschaft*

Die nachfolgende Abbildung 15 zeigt die Klasse *OrginäreQuantitativeEigenschaft* mit den jeweiligen Subklassen.

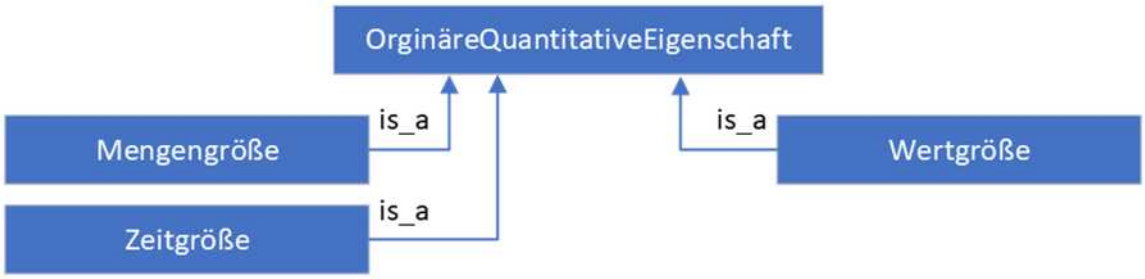


Abbildung 15: Subklassen der Klasse *OrginäreQuantitativeEigenschaft*

Die Klasse *OrginäreQuantitativeEigenschaft* stellt messbare Größeneinheiten dar. Die wesentliche Unterscheidung zur Klasse *DerivativeQuantitativeEigenschaft* besteht darin, dass die originäre quantitative Eigenschaft als eine beurteilungsfreie allgemeine Eigenschaft verstanden wird, beispielsweise als Mengen- oder Zeitgrößen.

Klasse	**Beschreibung**	**Subklasse von**
Mengengröße	Die Klasse *Mengengröße* dient als Oberklasse aller Mengengrößen. Die Mengengrößen geben eine messbare Einheit oder Größe einer definierten Eigenschaft an, der keine Zeit- oder Wertgröße zugrunde liegt.	OrginäreQuantitative Eigenschaft
Zeitgröße	Die Klasse *Zeitgröße* dient als Oberklasse aller Zeitgrößen. Die Zeitgröße beschreibt einen bestimmten Zeitpunkt, ein bestimmtes Datum oder ein Zeitintervall.	OrginäreQuantitative Eigenschaft
Wertgröße	Die Klasse *Wertgröße* dient als Oberklasse aller Wertgrößen. Die Wertgrößen beschreiben monetäre Werte.	OrginäreQuantitative Eigenschaft

Tabelle 13: Beschreibung der Subklassen der Klasse *OrginäreQuantitativeEigenschaft*

Die Abbildung 16 stellt die Klasse *ITSicherheitsEigenschaft* mit den zugehörigen Subklassen dar.

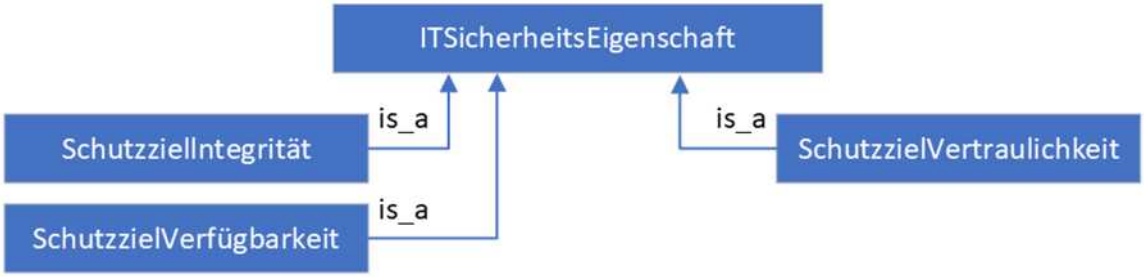


Abbildung 16: Subklassen der Klasse *ITSicherheitsEigenschaft*

IT-Sicherheits-Eigenschaften sind für ein sicherheitskritisches IT-System von elementarer Bedeutung und finden sich in den einschlägigen Leistungsbeschreibungen für sicherheitskritische IT-Systeme als zwingend zu erfüllende Anforderung wieder.

Um IT-Sicherheitseigenschaften zu definieren, eignen sich Schutzziele. Diese Schutzziele werden in der einschlägigen internationalen Fachliteratur als CIA-Triade (Confidentiality, Integrity and Availability) bezeichnet und gelten als grundlegende Eigenschaften der IT-Sicherheit; vgl. GONIWADA (2022), S. 374-375; LIEDTKE (2022), S. 19. Die Schutzziele Integrität, Verfügbarkeit und Vertraulichkeit sind grundlegende Ziele der IT-Sicherheit. Wichtig ist, zwischen den Schutzzielen, die nachfolgend als IT-Sicherheitseigenschaften definiert werden, und den Grundbedrohungen zu unterscheiden. Die Grundbedrohungen beschreiben potenzielle Gefahren für die Sicherheit von sicherheitskritischen IT-Systemen, wie zum Beispiel Datenverlust, Datendiebstahl oder unbefugten Zugriff auf Daten. Ein Schutzziel kann also als ein Ziel verstanden werden, das durch Sicherheitsmaßnahmen erreicht werden soll (somit als eine IT-Sicherheitseigenschaft), während die Grundbedrohung eine Bedrohung darstellt, die das Erreichen des Schutzziels gefährdet. Es ist daher wichtig, dass bei der Planung von Sicherheitsmaßnahmen sowohl die Grundbedrohungen als auch die Schutzziele berücksichtigt werden. Der festgestellte Schutzbedarf lässt sich nur schwer quantifizieren und beschränkt sich daher auf eine qualitative Beurteilung.

Die nachfolgende Tabelle 14 stellt die Subklassen der Klasse *ITSicherheitsEigenschaft* dar.

Klasse	**Beschreibung**	**Subklasse von**
Schutzziel Integrität	Die Klasse *SchutzzielIntegrität* dient als Oberklasse für alle Eigenschaften, welche das Schutzziel der Integrität betreffen.	ITSicherheits Eigenschaft

	Das Schutzziel „Integrität“ bezeichnet laut IT-Grundschutz das durchgängige Funktionieren von IT-Systemen sowie die Vollständigkeit und Richtigkeit von Daten.	
Schutzziel Verfügbarkeit	Die Klasse *SchutzzielVerfügbarkeit* beinhaltet als Oberklasse alle Eigenschaften, welche das Schutzziel der Verfügbarkeit betreffen. Das Schutzziel „Verfügbarkeit“ definiert, zu welchem Grad IT-Systeme, IT-Anwendungen, IT-Netzwerke und Daten einem Benutzer zur Verfügung stehen und ohne Einschränkung verwendet werden können.	ITSicherheits Eigenschaft
Schutzziel Vertraulichkeit	Die Klasse *SchutzzielVertraulichkeit* dient als Oberklasse für alle Eigenschaften, welche das Schutzziel der Vertraulichkeit betreffen. Das Schutzziel „Vertraulichkeit“ definiert den Schutz vor unbefugter Preisgabe von vertraulichen Daten. Vertrauliche Daten dürfen nur Befugten in der zulässigen Weise zugänglich sein.	ITSicherheits Eigenschaft

Tabelle 14: Beschreibung der Subklassen der Klasse *ITSicherheitsEigenschaft*

Die Abbildung 17 illustriert die Klasse *Systemumgebung* mit den dazugehörigen Subklassen.

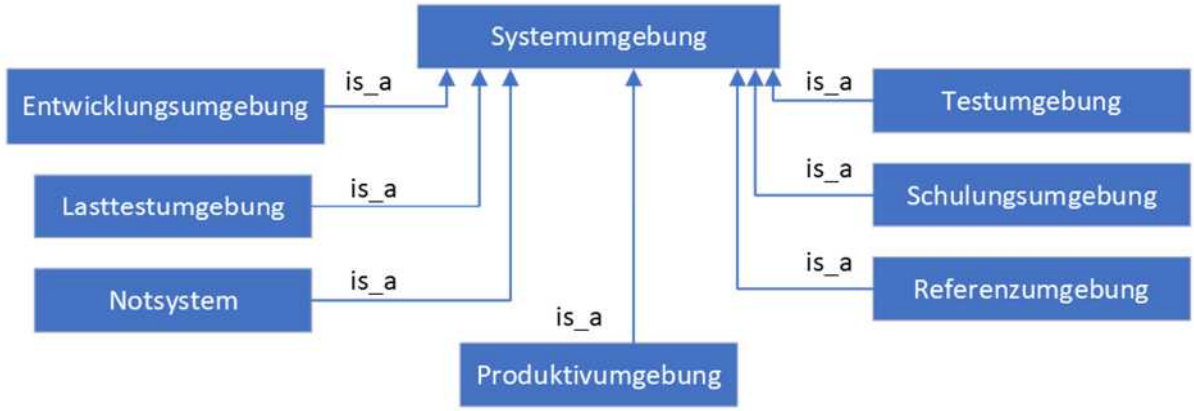


Abbildung 17: Subklassen der Klasse *Systemumgebung*

Systemumgebungen sind für sicherheitskritische IT-Systeme von zentraler Bedeutung, da sie für die Ausfallsicherheit sowie für das Testen von Software und Hardware von großer Bedeutung sind. Eine Anpassung an eine Produktivumgebung in Form einer Software-, Konfigurations- oder Hardwareanpassung wird auf verschiedenen Systemumgebungen zuvor getestet. Sobald sichergestellt ist, dass ein Ausfall des sicherheitskritischen IT-Systems unwahrscheinlich

ist, kann die Anpassung an die Produktivumgebung erfolgen. Das Durchlaufen der verschiedenen Systemumgebungen dient dazu, mögliche technische Risiken, wie sie beispielsweise durch Software oder Hardware entstehen können, frühzeitig auf den verschiedenen Systemumgebungen zu identifizieren, um diese für die Produktivumgebung auszuschließen. Die Klasse *Systemumgebung* wird in die Sublassen *Testumgebung*, *Lasttestumgebung*, *Notsystem*, *Referenzumgebung* und *Produktivumgebung* differenziert und in der nachfolgenden Tabelle15 als Subklasse der Klasse *Systemumgebung* näher erläutert.

Klasse	**Beschreibung**	**Subklasse von**
Entwicklungs umgebung	Die Klasse *Entwicklungsumgebung* dient als Oberklasse für alle Entwicklungsumgebungen. Unter Entwicklungsumgebung wird eine Systemumgebung verstanden, die hauptsächlich für die Entwicklung einer Software genutzt wird. Die Softwareentwickler einer Anwendung nutzen diese Systemumgebung, um beispielsweise Entwicklungstests durchzuführen.	Systemumgebung
Lasttest umgebung	Die Klasse *Lasttestumgebung* dient als Oberklasse für alle Lasttestumgebungen. Eine Lasttestumgebung ist eine Systemumgebung, die speziell für Lasttests eingerichtet wird. Das Ziel von Lasttests besteht darin, die maximale Auslastung eines IT-Systems zu ermitteln und die Verfügbarkeit des Systems unter Belastung zu überprüfen. Dabei wird nicht nur bei herkömmlichen IT-Systemen, sondern auch bei sicherheitskritischen IT-Systemen auf eine Leistungsfähigkeit und Ausfallsicherheit geachtet. Regelmäßige Lasttests sind notwendig, um auch bei stetiger Weiterentwicklung und Anpassung eines sicherheitskritischen IT-Systems die Auslastungsgrenze zu bestimmen. Da Lasttests die Leistung des Systems stark beeinträchtigen können, sollten sie nicht auf einer produktiven Umgebung durchgeführt werden. Stattdessen werden sie in der Lasttestumgebung durchgeführt, um den produktiven Betrieb des Systems nicht zu gefährden.	Systemumgebung
Notsystem	Die Klasse *Notsystem* dient als Oberklasse für alle Notsysteme.	Systemumgebung

	Ein Notsystem stellt eine Systemumgebung dar, die genutzt wird, wenn die Produktivumgebung ausgefallen ist. Die Aktivierung des Notsystems kann automatisiert oder manuell angestoßen werden. Ein Notsystem stellt eine Rückfallumgebung für den produktiven Betrieb eines sicherheitskritischen IT-Systems dar.	
Produktiv umgebung	Die Klasse *Produktivumgebung* dient als Oberklasse für alle Produktivumgebungen. Die Produktivumgebung ist die Systemumgebung, die durch die Nutzer eines sicherheitskritischen IT-Systems aktiv genutzt wird. Die Produktivumgebung ist die wichtigste Systemumgebung, da auf diese Umgebung der produktive Betrieb des sicherheitskritischen IT-Systems erfolgt. Der Ausfall der Produktivumgebung hätte einen Systemausfall zur Folge.	Systemumgebung
Referenz umgebung	Die Klasse *Referenzumgebung* dient als Oberklasse für alle Referenzumgebungen. Die Referenzumgebung ist die Systemumgebung, die als Referenz für die Produktivumgebung dient. Die Umgebung dient dazu, Softwaretests, die auf einer produktionsnahen Umgebung erfolgen sollen, bevor eine Software auf der produktiven Umgebung eingespielt wird, durchzuführen. In der Regel ist die Referenzumgebung im Aufbau ähnlich oder gleich zu einer Produktivumgebung.	Systemumgebung
Schulungs umgebung	Die Klasse *Schulungsumgebung* dient als Oberklasse für alle Schulungsumgebungen. Die Schulungsumgebung ist die Systemumgebung, in der im Rahmen von Schulungen die Benutzergruppen (z. B. Administratoren) geschult werden.	Systemumgebung
Testumgebung	Die Klasse *Testumgebung* dient als Oberklasse für alle Testumgebungen. Die Testumgebung ist die Systemumgebung, auf der übergreifende Software-, Daten- und Hardwaretests stattfinden.	Systemumgebung

Tabelle 15: Beschreibung der Subklassen der Klasse *Systemumgebung*

Die nachfolgende Abbildung18 zeigt die Klasse *Virtualisierung* mit den zugehörigen Subklassen.

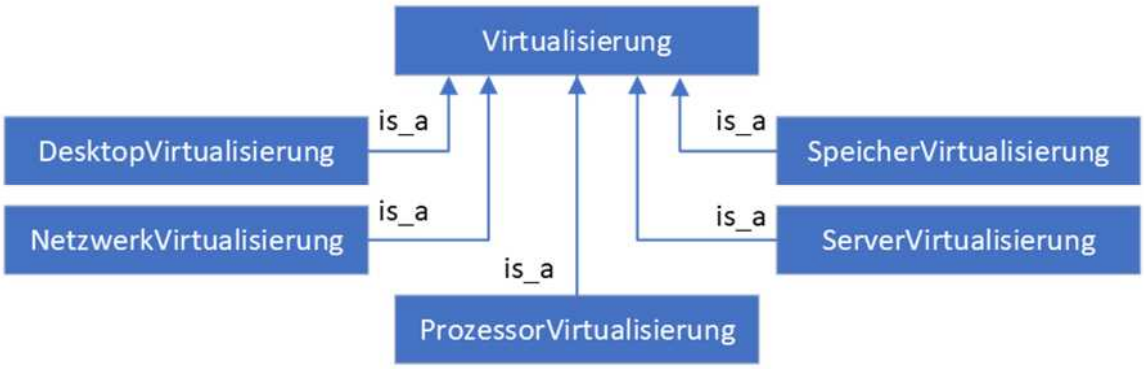


Abbildung 18: Subklassen der Klasse *Virtualisierung*

Die Virtualisierung von Hardware-Komponenten stellt in der Informationstechnologie eine gängige Bereitstellungsart von technischen Ressourcen dar. Die nachfolgende Tabelle16 erläutert die Subklassen der Klasse *Virtualisierung*, welche die sprachlichen Ausdrucksmittel bereitstellen, um die unterschiedlichen Virtualisierungskonzepte auszudrücken.

Klasse	**Beschreibung**	**Subklasse von**
Desktop Virtualisierung	Die Klasse *DesktopVirtualisierung* dient als Oberklasse für alle Arten der Desktop-Virtualisierung. Unter einer Desktop-Virtualisierung wird ein Virtualisierungskonzept verstanden, wonach ein Desktop Client virtuell bereitgestellt wird.	Virtualisierung
Netzwerk Virtualisierung	Die Klasse *NetzwerkVirtualisierung* dient als Oberklasse für alle Arten der Netzwerk-Virtualisierung. Unter einer Netzwerk-Virtualisierung wird ein Virtualisierungskonzept verstanden, wonach ein Netzwerk virtuell bereitgestellt wird.	Virtualisierung
Prozessor Virtualisierung	Die Klasse *ProzessorVirtualisierung* dient als Oberklasse für alle Arten der Prozessor-Virtualisierung. Unter einer Prozessor-Virtualisierung wird ein Virtualisierungskonzept verstanden, wonach ein Prozessor virtuell bereitgestellt wird.	Virtualisierung
Server Virtualisierung	Die Klasse *ServerVirtualisierung* dient als Oberklasse für alle Arten der Server-Virtualisierung.	Virtualisierung

	Unter einer Server-Virtualisierung wird ein Virtualisierungskonzept verstanden, wonach ein Server virtuell bereitgestellt wird.	
Speicher Virtualisierung	Die Klasse *SpeicherVirtualisierung* dient als Oberklasse für alle Arten der Speicher-Virtualisierung. Unter einer Speicher-Virtualisierung wird ein Virtualisierungskonzept verstanden, wonach ein Speicher virtuell bereitgestellt wird.	Virtualisierung

Tabelle16: Beschreibung der Subklassen der Klasse *Virtualisierung*

Die nachfolgende Abbildung19 stellt die Klasse *Recht* mit den dazugehörigen Subklassen dar.

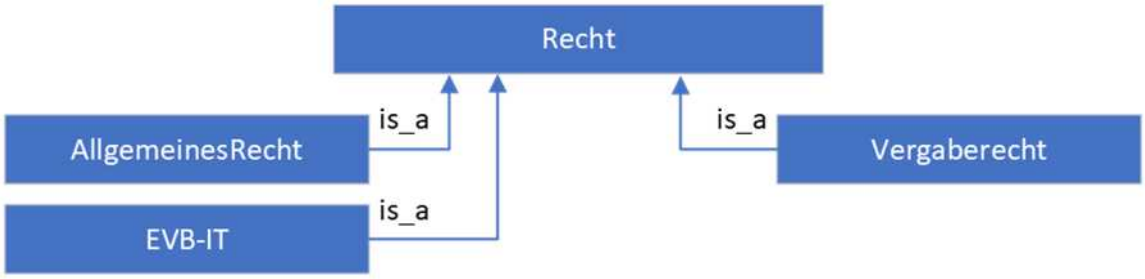


Abbildung 19: Subklassen der Klasse *Recht*

Die Subklassen der Klasse *Recht* repräsentieren unterschiedliche zentrale Belange des Rechts für sicherheitskritische IT-Projekte. Diese umfassen das Allgemeine Recht, die EVB-IT (Ergänzende Vertragsbedingungen für die Beschaffung von IT-Leistungen) und das Vergaberecht. Die Unterteilung ist darauf zurückzuführen, dass sich rechtliche Belange für ein sicherheitskritisches IT-Projekt zwischen der allgemeinen Gesetzgebung, dem Vertragsrecht für die Beschaffung von IT-Leistungen der Bundesrepublik Deutschland und dem Vergaberecht unterscheiden lassen. Die EVB-IT und das Vergaberecht sind Rechtsrahmen, die Auswirkungen auf die öffentliche Vergabe von IT-Leistungen haben und sich vom Allgemeinen Recht, das als Grundrahmen verstanden werden kann, unterscheiden.

Verträge zur Beschaffung von sicherheitskritischen IT-Systemen erfordern aufgrund der Komplexität der Beschaffung und des hohen Stellenwerts für die Gesellschaft eine umfassende Regelung in verschiedenen Bereichen, wie zum Beispiel bei der Beschreibung beiderseitiger Leistungsverpflichtungen und der Festlegung von Haftungsbedingungen bei Störungen oder Ausfällen solcher Systeme. Die gesetzlichen Regelungen zum Schuldrecht im Bürgerlichen Gesetzbuch (BGB) allein sind für diesen Anspruch nicht ausreichend. Obwohl sie Hinweise für ein mögliches Vertragswerk liefern, werden IT-spezifische Aspekte nicht angemessen berücksichtigt.

Demzufolge müssen als Ergänzung zu den genannten bestehenden Gesetzesgrundlagen Rahmenbedingungen geschaffen werden, um solche komplexe Vorhaben auf vertragliche Grundlagen zu stellen. Daher erarbeitete das Bundesministerium des Innern diese ergänzenden Vertragsbedingungen für die Informationstechnologie (EVB-IT) hinsichtlich des Einkaufs von öffentlichen IT-Leistungen. Die Subklasse *EVB-IT* subsumiert die unterschiedlichen EVB-IT-Vertragstypen, die für verschiedene Anwendungsgebiete angewendet werden können.

Das Vergaberecht wurde bereits bei der Klasse *VergabeverfahrensArt* angerissen, da sicherheitskritische IT-Projekte, welche für die Anwendung bei Behörden und Organisationen mit Sicherheitsaufgaben (BOS) bestimmt sind, dem öffentlichen Vergaberecht unterliegen. Das Vergabeverfahren kann als die Anwendung des Vergaberechts verstanden werden. Das Vergaberecht legt mit seinen Regelungen fest, wie Bund, Länder und Kommunen vorgehen müssen, um Güter am Markt einzukaufen oder Bau- und Dienstleistungen in Auftrag zu geben. Es soll sicherstellen, dass Haushaltsmittel der öffentlichen Auftraggeber wirtschaftlich und in einem wettbewerblichen, transparenten und nicht-diskriminierenden Vergabeverfahren eingesetzt werden, um dem wirtschaftlichsten Angebot, gemessen am Preis-Leistungs-Verhältnis, den Vorzug zu geben.

In der nachfolgenden Tabelle17 werden die Subklassen der Klasse *Recht* erläutert.

Klasse	**Beschreibung**	**Subklasse von**
Allgemeines Recht	Die Klasse *AllgemeinesRecht* umfasst alle allgemein geltenden Gesetze.	Recht
EVB-IT	Die Klasse *EVB-IT* ist die Oberklasse für die unterschiedlichen Vertragsbedingungen der EVB-IT.	Recht
Vergaberecht	Die Klasse *Vergaberecht* repräsentiert die Oberklasse des Vergaberechts, welche das aktuelle und das alte Vergaberecht der Bundesrepublik Deutschland subsumiert.	Recht

Tabelle 17: Beschreibung der Subklassen der Klasse *Recht*

Die nachfolgende Abbildung20 zeigt die Subklassen der Klasse *Zuschlagskriterium*.

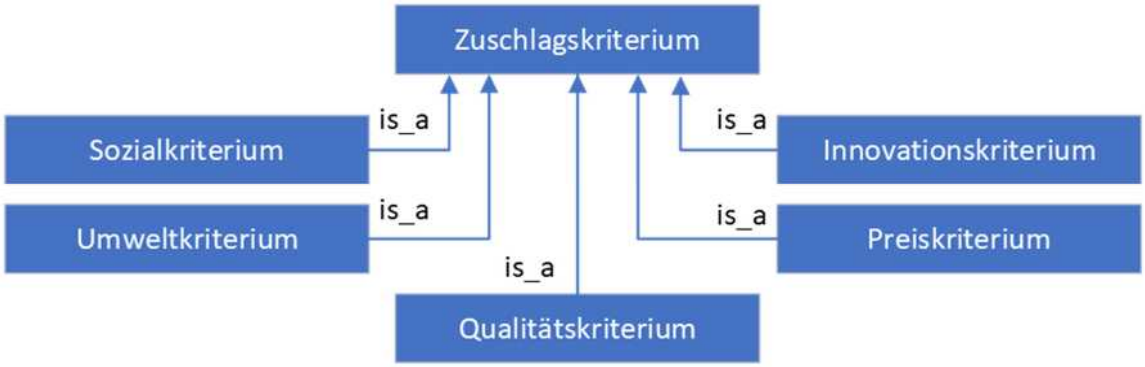


Abbildung 20: Subklassen der Klasse *Zuschlagskriterium*

Der Zuschlag wird gemäß § 127 GWB (Gesetz gegen Wettbewerbsbeschränkungen) für das wirtschaftlichste Angebot erteilt. Die Ermittlung des wirtschaftlichsten Angebots erfolgt in der Regel auf Grundlage eines Preis-Leistungs-Verhältnisses. In § 58 VgV (Vergabeverordnung) „Zuschlag und Zuschlagskriterien" werden neben dem Preis auch qualitative, innovative, umweltbezogene und soziale Kriterien als Zuschlagskriterien genannt. Dies begründet auch die Auswahl der in Tabelle18 erläuterten Klassen als Subklassen der Klasse *Zuschlagskriterien*. Die Zuschlagskriterien stellen für die Vergabe eines öffentlichen Auftrags eine zentrale Bedeutung dar und sind daher auch Gegenstand verschiedener Nachprüfungsanträge. Die Zuschlagskriterien müssen in Verbindung mit dem Auftragsgegenstand stehen.

Klasse	**Beschreibung**	**Subklasse von**
Sozial kriterium	Die Klasse *Sozialkriterium* repräsentiert die Oberklasse für alle Zuschlagskriterien, die soziale Kriterien berücksichtigen. Die Vergabeverordnung (VgV) beschreibt soziale Kriterien als die „Zugänglichkeit der Leistung insbesondere für Menschen mit Behinderungen" und Kriterien, die bestimmte Aspekte des gesellschaftlichen Zusammenlebens betreffen."	Zuschlags kriterium
Umwelt kriterium	Die Klasse *Umweltkriterium* repräsentiert die Oberklasse für alle Zuschlagskriterien, denen umweltbezogene Kriterien zugrunde liegen. Umweltbezogene Zuschlagskriterien sind Kriterien, die an Merkmale der Leistung anknüpfen, die den Effekt der Leistung auf die Umwelt abbilden.	Zuschlags kriterium

Qualitäts kriterium	Die Klasse *Qualitätskriterium* repräsentiert die Oberklasse für alle Zuschlagskriterien, die auf qualitätsbezogenen Kriterien basieren. Qualitätskriterien sind solche Kriterien, die auf die Qualität der ausgeschriebenen Leistung Einfluss haben können. Die VgV beschreibt unter anderem, dass die Qualifikation und Erfahrung des für die Ausführung des Auftrags verantwortlichen Personals Einfluss auf die Qualität der Auftragsausführung haben können. Des Weiteren sind auch die Verfügbarkeit der Leistung und technischer Hilfe sowie Lieferbedingungen wie Liefertermin, Lieferverfahren sowie Liefer- und Ausführungsfristen unter Qualitätskriterien zu fassen.	Zuschlags kriterium
Preis kriterium	Die Klasse *Preiskriterium* stellt die übergeordnete Klasse für alle Zuschlagskriterien dar, die auf preisbezogenen Kriterien basieren. Preisbezogene Zuschlagskriterien sind Kriterien, die an den Preis einer Leistung anknüpfen.	Zuschlags kriterium
Innovations kriterium	Die Klasse *Innovationskriterium* stellt die übergeordnete Klasse für alle Zuschlagskriterien dar, die auf innovationsbezogenen Kriterien basieren.	Zuschlags kriterium

Tabelle 18: Beschreibung der Subklassen der Klasse *Zuschlagskriterium*

Neben der Klasse *Eigenschaft* ist die Klasse *Objekt* eine zentrale Klasse auf der ersten Hierarchieebene der sicherheitskritischen IT-Projekt-Ontologie. Diese erste Hierarchieebene wurde bereits in der zugrunde gelegten PM-Domänen-Ontologie festgelegt. Nachfolgend wird auf exemplarisch ausgewählte Subklassen der Klasse *Objekt* näher eingegangen.

Die Klasse *Objekt* wird in die Subklassen *KognitivesObjekt* und *RealesObjekt* durch die verwendete PM-Domänen-Ontologie unterteilt, wie die nachfolgende Abbildung 21 illustriert.

Abbildung 21: Subklassen der Klasse *Objekt*

Die Unterscheidung in die Subklassen *KognitivesObjekt* und *RealesObjekt* wird in der sicherheitskritischen IT-Projekt-Ontologie als Differenzierung zwischen Denk- und Erfahrungsob-

jekten beschrieben. Kognitive Objekte sind Objekte, die durch die „innere Wahrnehmung“ eines beliebigen Akteurs entstehen, unabhängig davon, ob die Objekte in der Realität existieren. Bei realen Objekten handelt es sich dagegen um Inhalte von Erfahrungsprozessen. Die realen Objekte existieren in der jeweils konzipierten Domäne und können beobachtet werden.

Die nachfolgende Tabelle19 erläutert die Subklassen der Klasse *Objekt*.

Klasse	**Beschreibung**	**Subklasse von**
KognitivesObjekt	Die Klasse *KognitivesObjekt* ist die Oberklasse aller kognitiven Objekte. Die Subklassen der Klasse *KognitivesObjekt* repräsentieren Objekte von Denkprozessen, welche durch die „innere Wahrnehmung“ entstehen. Die kognitiven Objekte müssen nicht zwangsläufig in der Realität existieren.	Objekt
RealesObjekt	Die Klasse *RealesObjekt* ist die Oberklasse aller realen Objekte. Die Subklassen der Klasse *RealesObjekt* repräsentieren Objekte von Erfahrungsprozessen, welche durch die „äußere Wahrnehmung“ entstehen. Reale Objekte existieren in der konstruierten Domäne und können von Akteuren mittels ihrer Sinne beobachtet werden.	Objekt

Tabelle 19: Beschreibung der Subklassen der Klasse *Objekt*

Die nachfolgende Abbildung 22 stellt die Klasse *System* mit den dazugehörigen Subklassen dar.

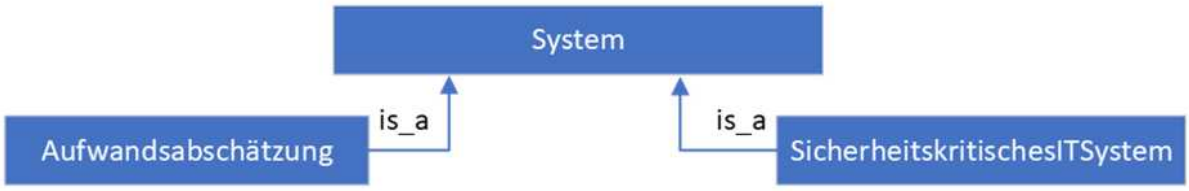


Abbildung 22: Subklassen der Klasse *System*

Die nachfolgende Tabelle20 erläutert die die Subklassen der Klasse *System*.

Klasse	**Beschreibung**	**Subklasse von**
Aufwands abschätzung	Die Klasse *Aufwandsabschätzung* stellt die Oberklasse für alle Arten der Aufwandsabschätzung dar. Es gibt verschiedene Arten der Aufwandsabschätzung, die durch diese Klasse abgedeckt werden.	System
Sicherheits kritisches ITSystem	Die Klasse *SicherheitskrtischesITSystem* ist die Oberklasse für alle Arten von sicherheitskritischen IT-Systemen.	System

Tabelle 20: Beschreibung der Subklassen der Klasse *System*

Die nachfolgende Abbildung 23 zeigt die Klasse *Urteil* mit den dazugehörigen Subklassen.

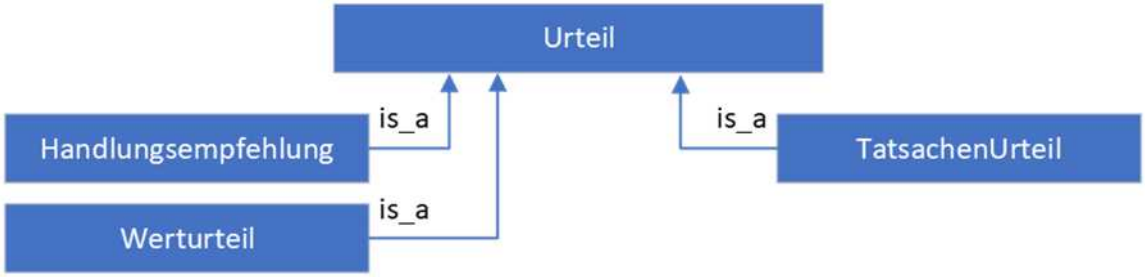


Abbildung 23: Subklassen der Klasse *Urteil*

Die Klasse *Urteil* wird in die Klassen *Handlungsempfehlung*, *Werturteil* und *TatsachenUrteil* unterteilt. Die nachfolgende Tabelle21 erläutert die Subklassen der Klasse *Urteil*.

Klasse	**Beschreibung**	**Subklasse von**
Handlungs empfehlung	Die Klasse *Handlungsempfehlung* ist die Oberklasse für alle Arten von Handlungsempfehlungen. Handlungsempfehlungen werden charakterisiert als Vorschläge für Handlungspläne, welche nicht zwingend einzuhalten sind. Sie dienen als Empfehlungen für eine Vorgehensweise. Handlungsempfehlungen werden beispielsweise von Softwareanbietern oder öffentlichen Institutionen herausgegeben, um eine Vorgehensweise zu unterstützen.	Urteil
Werturteil	Die Klasse *Werturteil* ist die Oberklasse für alle Arten von Werturteilen.	Urteil

	Werturteile werden als persönliche Stellungnahmen charakterisiert, in denen ein Sachverhalt positiv oder negativ ausgezeichnet wird. Werturteile sind im Gegensatz zu Tatsachenaussagen nicht empirisch überprüfbar.	
Tatsachen Urteil	Die Klasse *TatsachenUrteil* ist die Oberklasse für alle Arten von gerichtlichen Urteilen.	Urteil

Tabelle 21: Beschreibung der Subklassen der Klasse *Urteil*

Die nachfolgende Abbildung24 illustriert die Klasse *Wissen* mit den dazugehörigen Subklassen.

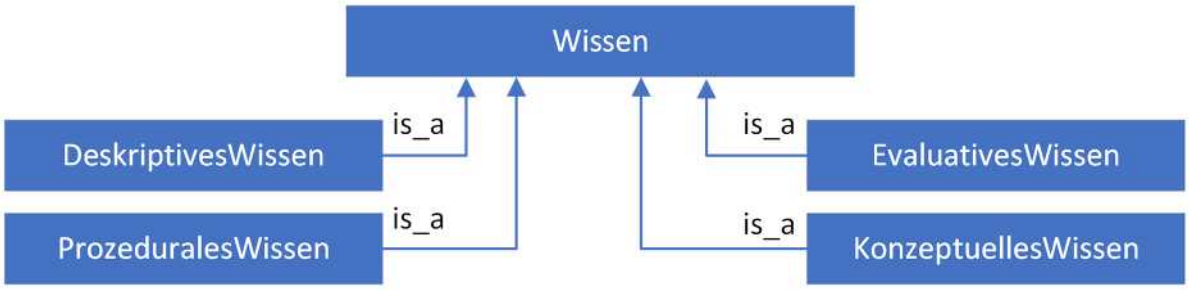


Abbildung 24: Subklassen der Klasse *Wissen*

Für die sicherheitskritische IT-Projekt-Ontologie werden die Klassen *DeskriptivesWissen*, *EvaluativesWissen*, *KonzeptuellesWissen* und *ProzeduralesWissen* verwendet. Die nachfolgende Tabelle22 erläutert die genannten Subklassen.

Klasse	**Beschreibung**	**Subklasse von**
Deskriptives Wissen	Die Klasse *DeskriptivesWissen* ist die Oberklasse für alle Arten des deskriptiven Wissens. Deskriptives Wissen ist definiert als das Wissen über Tatsachen, welche durch Theorien, Konzepte, Prinzipien, Schemata und Ideen ausgedrückt wird.	Wissen
Prozedurales Wissen	Die Klasse *ProzeduralesWissen* ist die Oberklasse für alle Arten von prozeduralem Wissen. „Prozedurales Wissen“ beschreibt das praktisch anwendbare Handlungswissen und die Fähigkeit, deklaratives Wissen miteinander zu verknüpfen und in Handlungsabläufen anzuwenden.	Wissen
Evaluatives Wissen	Die Klasse *EvaluativesWissen* ist die Oberklasse für alle Arten von evaluativem Wissen.	Wissen

	„Evaluatives Wissen“ repräsentiert Wissen, welches in Form einer Bewertung vorliegt.	
Konzeptuelles Wissen	Die Klasse *KonzeptuellesWissen* ist die Oberklasse für alle Arten von konzeptionellem Wissen. „Konzeptuelles Wissen“ ist die Basis für ein vertieftes Verständnis von fachspezifischen Inhalten. Es wird definiert als das Verständnis von Konzepten, die eine Domäne beeinflussen und sich in einen Zusammenhang stellen lassen. Das konzeptuelle Wissen umfasst Erklärungsmechanismen für ganz spezifische Sachverhalte, die miteinander verknüpft werden müssen.	Wissen

Tabelle 22: Beschreibung der Subklassen der Klasse *Wissen*

Die nachfolgende Abbildung25 zeigt die Klasse *DeskriptivesWissen* mit den dazugehörigen Subklassen.

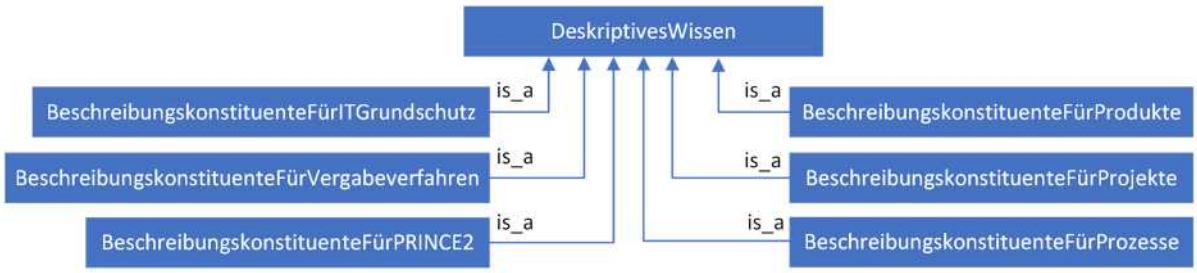


Abbildung 25: Subklassen der Klasse *DeskriptivesWissen*

Die nachfolgende Tabelle23 erläutert die Subklassen der Klasse *DeskriptivesWissen*.

Klasse	**Beschreibung**	**Subklasse von**
Beschreibungs konstituente FürITGrundschutz	Die Klasse *BeschreibungskonstituenteFür ITGrundschutz* dient als Oberklasse, welche die sprachlichen Ausdrucksmittel bereitstellt, um den IT-Grundschutz auszudrücken.	Deskriptives Wissen
Beschreibungs konstituente FürVergabeverfahren	Die Klasse *Beschreibungskonstituente FürVergabeverfahren* ist die Oberklasse, welche die sprachlichen Ausdrucksmittel bereitstellt, um die Vergabeschritte auszudrücken.	Deskriptives Wissen
Beschreibungs konstituente-FürPRINCE2	Die Klasse *Beschreibungskonstituente FürPRINCE2* fungiert als übergeordnete Klasse, welche sämtliche Arten von Beschreibungskonstituenten der Projektmanagementmethode PRINCE2	Deskriptives Wissen

	umfasst. In dieser Hinsicht bietet die Klasse die sprachlichen Ausdrucksmittel, die für die Beschreibung der festgelegten Begriffe in der Projektmanagementmethode PRINCE2 erforderlich sind.	
Beschreibungs konstituenteFür Produkte	Die Klasse *BeschreibungskonstituentenFürProdukte* dient als Oberklasse für alle Arten von Beschreibungskonstituenten, die für Produkte relevant sind. Sie bietet sprachliche Ausdrucksmittel, um die Anforderungen an Realgüter (ausgedrückt mit der Klasse *Realgut*) zu beschreiben.	Deskriptives Wissen
Beschreibungs konstituenteFür Projekte	Die Klasse *Beschreibungskonstituente FürProjekte* ist die Oberklasse für sämtliche Arten von Beschreibungskonstituenten im Kontext von Projekten. Sie stellt die sprachlichen Ausdrucksmittel bereit, welche zur Beschreibung von Projekten verwendet werden und somit eine einheitliche und standardisierte Beschreibung von Projekten ermöglichen.	Deskriptives Wissen
Beschreibungs konstituenteFür Prozesse	Die Klasse *Beschreibungskonstituente FürProzesse* ist die Oberklasse für alle Arten von Beschreibungskonstituenten im Kontext von Prozessen. Die Klasse stellt sprachliche Ausdrucksmittel zur Verfügung, die Maßnahmen beschreiben und somit eine umfassende Beschreibung von Prozessen ermöglichen.	Deskriptives Wissen

Tabelle 23: Beschreibung der Subklassen der Klasse *DeskriptivesWissen*

Die nachfolgende Abbildung26 stellt die Klasse *EvaluativesWissen* mit den zugehörigen Subklassen dar.

Abbildung 26: Subklassen der Klasse *EvaluativesWissen*

Die nachfolgende Tabelle24 erläutert die Subklassen der Klasse *EvaluativesWissen*.

Klasse	**Beschreibung**	**Subklasse von**
Bewertungs konstituente FürProdukte	Die Klasse *BewertungskonstituenteFürProdukte* fungiert als übergeordnete Klasse für sämtliche Arten von Bewertungen im Kontext von Produkten. Diese Klasse stellt sprachliche Ausdrucksmittel bereit, um die Erfüllung der Anforderungen zu bewerten.	EvaluativesWissen
Bewertungs konstituente FürProjekte	Die Klasse *BewertungskonstituenteFürProjekte* fungiert als Oberklasse für alle Bewertungen im Zusammenhang mit Projekten. Diese Klasse bietet die sprachlichen Ausdrucksmittel zur Bewertung von Projekten.	EvaluativesWissen

Tabelle 24: Beschreibung der Subklassen der Klasse *EvaluativesWissen*

Die nachfolgende Abbildung27 zeigt die Klasse *KonzeptuellesWissen* mit der zugehörigen Subklasse.

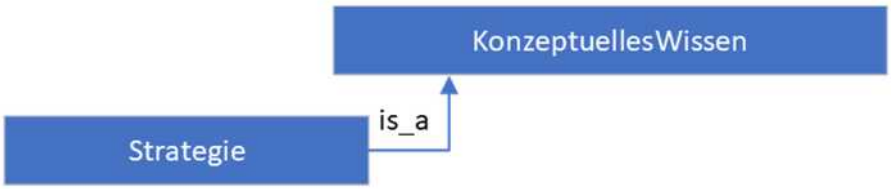


Abbildung 27: Subklasse der Klasse *KonzeptuellesWissen*

Die nachfolgende Tabelle25 erläutert die Subklasse der Klasse *KonzeptuellesWissen*.

Klasse	**Beschreibung**	**Subklasse von**
Strategie	Die Klasse *Strategie* ist die Oberklasse für alle Arten von Strategien.	KonzeptuellesWissen

Tabelle 25: Beschreibung der Subklasse der Klasse *KonzeptuellesWissen*

Die nachfolgende Abbildung 28 illustriert die Klasse *BeschreibungskonstituenteFürPRINCE2* mit den zugehörigen Subklassen.

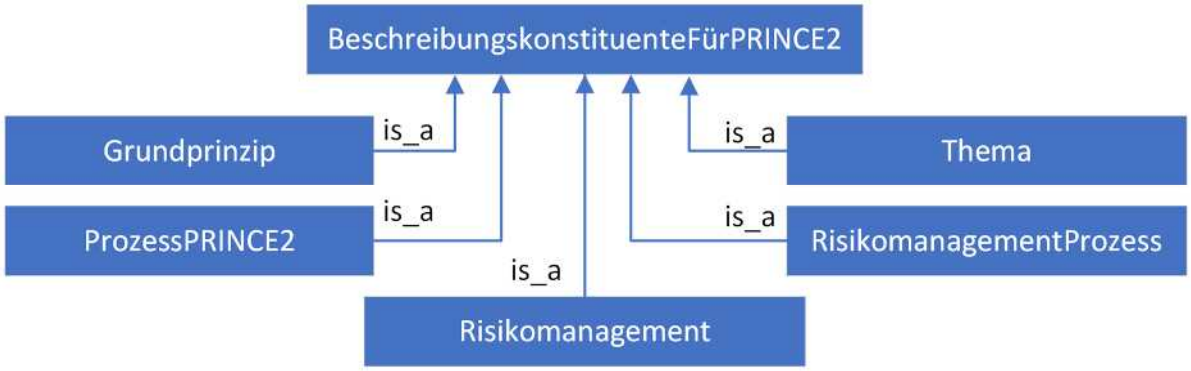


Abbildung 28: Subklassen der Klasse *BeschreibungskonstituenteFürPRINCE2*

Die nachfolgende Tabelle26 erläutert die Subklassen der Klasse *Beschreibungskonstituente-FürPRINCE2*.

Klasse	**Beschreibung**	**Subklasse von**
Grundprinzip	Die Klasse *Grundprinzip* bildet die Oberklasse für die Grundprinzipien von PRINCE2. Diese Grundprinzipien stellen die essenziellen Leitprinzipien dar, die während der Durchführung eines PRINCE2-Projekts beachtet werden müssen.	Beschreibungskonstituente FürPRINCE2
ProzessPRINCE2	Die Klasse *ProzessPRINCE2* stellt die übergeordnete Klasse dar, die die Prozesse von PRINCE2 repräsentiert. Diese Prozesse bestehen aus einer strukturierten Abfolge von Vorgängen, die darauf ausgerichtet sind, ein definiertes Ziel zu erreichen.	Beschreibungskonstituente FürPRINCE2
Risikomanagement	Die Klasse *Risikomanagement* repräsentiert die Oberklasse für alle Arten des Risikomanagements. Anmerkung: Da es sich um eine polymorphe Subsumierung handelt, werden in der Spalte rechts zwei übergeordnete Klassen angegeben. Das Risikomanagement umfasst sämtliche Maßnahmen zur Identifikation von Risiken sowie zur Bewältigung der mit diesen Risiken verbundenen Vorgänge in Projekten.	Beschreibungskonstituente FürPRINCE2 Methode
Risikomanagement Prozess	Die Klasse *RisikomanagementProzess* ist die Oberklasse für alle Arten von Risikomanagementprozessen von PRINCE2.	Beschreibungskonstituente FürPRINCE2

	Die Prozesse des Risikomanagements von PRINCE2 repräsentieren eine strukturierte Abfolge von Aktivitäten, welche die Identifikation, Bewertung, Planung, Implementierung von Gegenmaßnahmen und Kommunikation von Risiken beinhalten.	
Thema	Die Klasse *Thema* fungiert als Oberklasse für alle Themen von PRINCE2, welche wesentliche Bestandteile des Projektmanagements darstellen und im Laufe des Projektlebenszyklus kontinuierlich behandelt werden müssen.	Beschreibungs konstituente Für PRINCE2

Tabelle 26: Beschreibung der Subklassen der Klasse *BeschreibungskonstituenteFürPRINCE2*

Die nachfolgende Abbildung29 stellt die Klasse *BeschreibungskonstituenteFürScrum* mit den zugehörigen Subklassen dar.

Abbildung 29: Subklassen der Klasse *BeschreibungskonstituenteFürScrum*

Die Unterteilung der Klasse orientiert sich an der Unterscheidung zwischen Ereignissen und Artefakten im Scrum-Guide von SCHWABER/SUTHERLAND (2020), S. 7-12, der als Standardliteratur für Scrum gilt.

Klasse	**Beschreibung**	**Subklasse von**
ScrumEreignis	Die Klasse *ScrumEreignis* stellt die Oberklasse für alle Scrum-Ereignisse wie beispielsweise Daily, Sprint und Sprint Retrospektive dar.	Beschreibungs konstituente FürScrum
ScrumArtefakt	Die Klasse *ScrumArtefakt* stellt die Oberklasse für alle Scrum-Artefakte wie beispielsweise Produkt-Inkrement, Produkt-Backlog und Sprint-Backlog dar.	Beschreibungs konstituente FürScrum

Tabelle 27: Beschreibung der Subklassen der Klasse *BeschreibungskonstituenteFürScrum*

Die nachfolgende Abbildung30 stellt die Klasse *BeschreibungskonstituenteFürProdukte* mit den zugehörigen Subklassen dar.

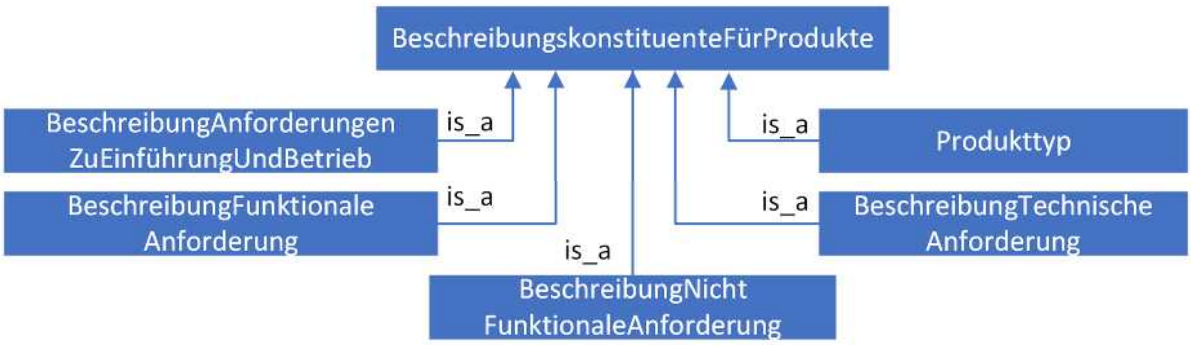


Abbildung 30: Subklassen der Klasse *BeschreibungskonstituenteFürProdukte*

Die Unterteilung der Klasse *BeschreibungskonstituenteFürProdukte* orientiert sich an den beschriebenen Anforderungen aus den in diesem Beitrag zugrunde gelegten Leistungsbeschreibungen. Die Leistungsbeschreibungen differenzieren die Anforderungen an das hybride Leistungsbündel in die nachfolgenden Bereiche:

- Anforderungen zu Einführung und Betrieb
- funktionale Anforderungen
- nicht-funktionale Anforderungen
- technische Anforderungen

Die durch die Leistungsbeschreibungen vorgenommene Unterteilung der Anforderungen eignet sich auch für die weitergehende Differenzierung der Klasse *BeschreibungskonstituenteFürProdukte*. Die nachfolgende Tabelle 28 ordnet die Anforderungsbereiche an das hybride Leistungsbündel aus den Leistungsbeschreibungen den jeweiligen Klassenbezeichnungen der Subklassen der Klasse *BeschreibungskonstituenteFürProdukte* zu.

Anforderungsbereich	**Klassenbezeichnung**
Anforderungen zu Einführung und Betrieb	*BeschreibungAnforderungen ZuEinführungUndBetrieb*
Funktionale Anforderung	*BeschreibungFunktionaleAnforderung*
Nicht-Funktionale Anforderung	*BeschreibungNichtFunktionale Anforderung*
Technische Anforderung	*BeschreibungTechnischeAnforderung*

Tabelle 28: Klassenbezeichnungen der Anforderungsbereiche

Der Vorteil der vorgenommenen Differenzierung ist, dass eine praxisnahe sicherheitskritische IT-Projekt-Ontologie geschaffen wird, da die Differenzierung an den zugrunde gelegten Leistungsbeschreibungen angelehnt ist.

Neben den zuvor genannten Differenzierungen für die Klasse *BeschreibungskonstituenteFür Produkte* ist durch die zugrunde gelegte PRINCE2- und Risikomanagement-Ontologie die Subklasse *Produkttyp* vorgegeben. Die Klasse *Produkttyp* wurde durch eine polymorphe Subsumierung sowohl der Klasse *BeschreibungskonstituenteFürPRINCE2* als auch der Klasse *BeschreibungskonstituenteFürProdukte* untergeordnet. Diese polymorphe Subsumierung begründet sich dahingehend, dass die Klasse *Produkttyp* sowohl als Beschreibungskonstituente von Produkten im Allgemeinen als auch als Beschreibungskonstituente der Projektmanagementmethode PRINCE2 im Speziellen semantisch gleich verwendet wird. Im PRINCE2 ist der Produktbegriff von zentraler Bedeutung, um zwischen Spezialistenprodukten (welche den konkreten Liefergegenstand, wie z. B. ein sicherheitskritisches IT-System, beschreiben) und Managementprodukten (welche für die Erstellung der Spezialistenprodukte relevant sind) zu unterscheiden.

Die nachfolgende Tabelle29 erläutert die Subklassen der Klasse *Beschreibungskonstituente FürProdukte*.

Klasse	**Beschreibung**	**Subklasse von**
Beschreibung Anforderunngen ZuEinführung UndBetrieb	Die Klasse *BeschreibungAnforderungen ZuEinführungUndBetrieb* ist die Oberklasse für Anforderungen zur Einführung und zum Betrieb.	Beschreibungs konstituente FürProdukte
Beschreibung Funktionale Anforderung	Die Klasse *BeschreibungFunktionaleAnforderung* ist die Oberklasse für funktionale Anforderungen.	Beschreibungs konstituente FürProdukte
Beschreibung NichtFunktionale Anforderung	Die Klasse *BeschreibungNichtFunktionaleAnforderung* ist die Oberklasse für nicht-funktionale Anforderungen.	Beschreibungs konstituente FürProdukte
Beschreibung Technische Anforderung	Die Klasse *BeschreibungTechnischeAnforderung* ist die Oberklasse für technische Anforderungen.	Beschreibungs konstituente FürProdukte
Produkttyp	Die Klasse *Produkttyp* ist die Oberklasse für alle Produkttypen von PRINCE2. Die Klasse *Produkttyp* drückt den Produktbegriff von PRINCE2 aus und unterscheidet zwischen Management- und Spezialistenprodukten.	Beschreibungs konstituente FürProdukte Beschreibungs konstituente FürPRINCE2

Tabelle 29: Beschreibung der Subklassen der Klasse *BeschreibungskonstituenteFürProdukte*

An dieser Stelle wird exemplarisch die polymorphe Subsumierung mittels Protégé anhand der Klasse *Produkttyp* erläutert. Eine polymorphe Subsumierung lässt sich in Protégé durch den Class Expression Editor anlegen. Der Aufruf des Class Expression Editors erfolgt in der Klasse, in der die polymorphe Subsumierung erfolgen soll, und zwar mittels der Funktion „SubClass Of“, der Aufruf des Class Expression Editor erfolgt über das „Plus“-Symbol, das in der nachfolgenden Abbildung31 rot eingekreist dargestellt wird.

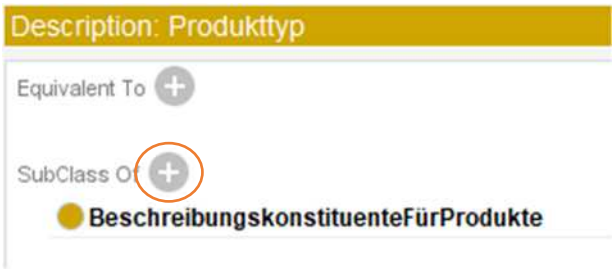


Abbildung 31: Aufruf der Funktion „SubClass Of“ in Protégé

Im Class Expression Editor können die übergeordneten Klassen durch die Eingabe der Klassennamen sowie durch den Einsatz des logischen Operators „and“ bestimmt werden. Nach Bestätigung im Class Expression Editor wird die Subklasse beiden Klassen als übergeordneten Klassen zugeordnet. Die nachfolgende Abbildung32 zeigt dies beispielhaft anhand der Klasse *Produkttyp* durch den Class Expression Editor.

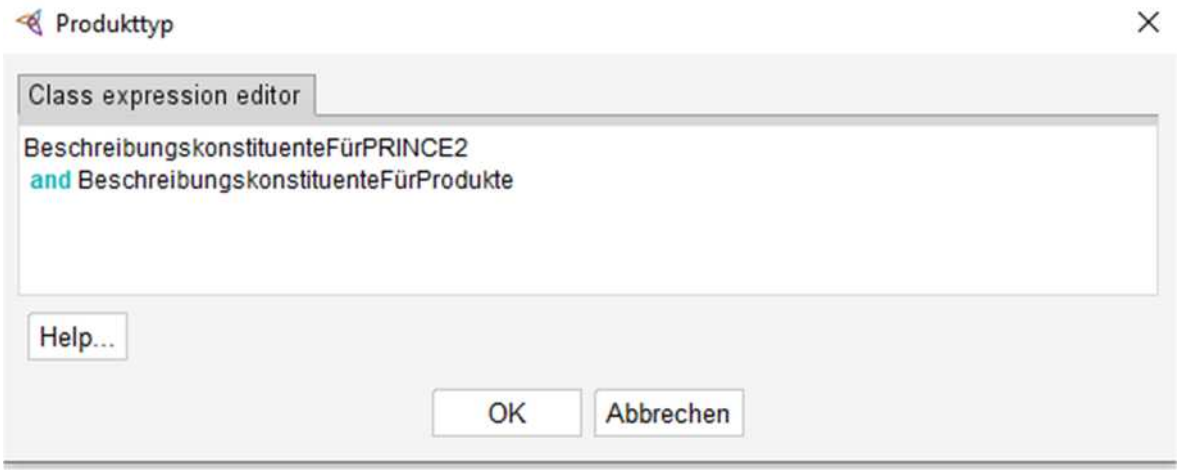


Abbildung 32: Class Expression Editor

Die nachfolgende Abbildung33 zeigt die Klasse *BeschreibungskonstituenteFürProjekte* mit den zugehörigen Subklassen.

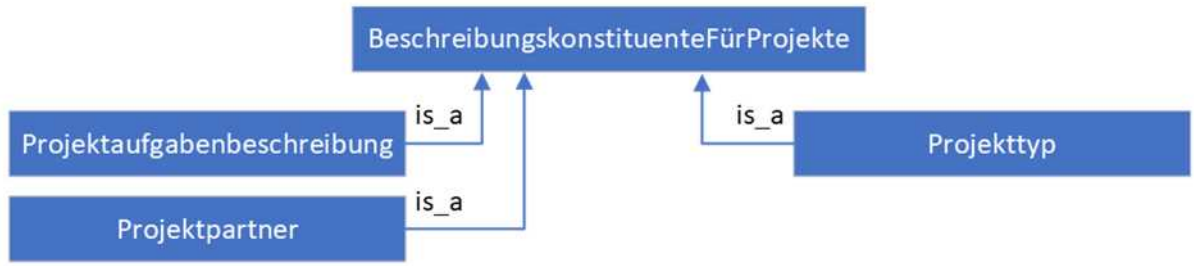


Abbildung 33: Subklassen der Klasse *BeschreibungskonstituenteFürProjekte*

Die nachfolgende Tabelle 30 erläutert die Subklassen der Klasse *Beschreibungskonstituente-FürProjekte*.

Klasse	**Beschreibung**	**Subklasse von**
Projektaufgaben beschreibung	Die Klasse *Projektaufgabenbeschreibung* ist die Oberklasse für alle Arten der Projektaufgaben-beschreibungen.	Beschreibungs konstituente FürProjekte
Projektpartner	Die Klasse *Projektpartner* ist die Oberklasse für alle Arten der Projektpartner. Die Projektpartner charakterisieren eine Personen-gruppe, die einen unmittelbaren Bezug zu einem sicherheitskritischen IT-Projekt besitzen.	Beschreibungs konstituente FürProjekte Stakeholder
Projekttyp	Die Klasse *Projekttyp* ist die Oberklasse für alle Arten von Projekttypen. Ein Projekttyp dient dazu, ein Projekt anhand bestimmter Merkmale zu charakterisieren.	Beschreibungs konstituente FürProjekte

Tabelle 30: Beschreibung der Subklassen der Klasse *BeschreibungskonstituenteFürProjekte*

Die nachfolgende Abbildung34 stellt die Klasse *BewertungskonstituenteFürProdukte* mit den zugehörigen Subklassen dar.

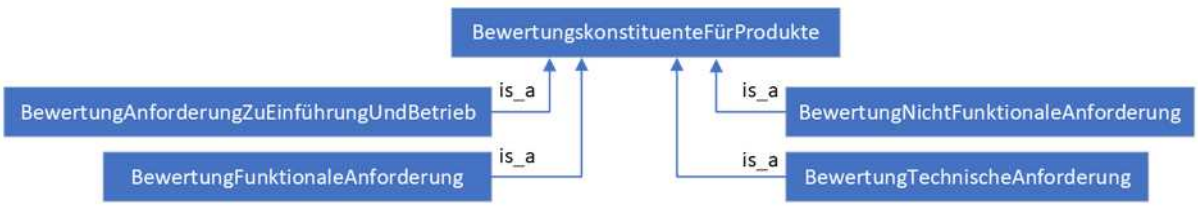


Abbildung 34: Subklassen der Klasse *BewertungskonstituenteFürProdukte*

Die Klasse *BewertungskonstituenteFürProdukte* stellt die sprachlichen Ausdrucksmittel für das Bewertungswissen von Anforderungen eines sicherheitskritischen IT-Projekts bereit. Die Differenzierung dieser Klasse erfolgt analog zur Differenzierung der Klasse *BeschreibungskonstituenteFürProdukte*, die sich an den Anforderungsbereichen der zugrunde liegenden Leistungsbeschreibungen orientiert. Das Ziel dieser Klasse ist es, die beschriebenen Anforderungen aus der Klasse *BeschreibungskonstituenteFürProdukte* in der Klasse *BewertungskonstituenteFürProdukte* als bewertendes Wissen zu repräsentieren.

Durch die Konstruktionsentscheidung, die Anforderungen einer Leistungsbeschreibung in der Klasse *BeschreibungskonstituenteFürProdukte* als rein beschreibendes Wissen zu repräsentieren und mittels nicht-taxonomischer Relationen mit der Klasse *BewertungskonstituenteFürProdukte* zu verknüpfen, die das bewertende Wissen repräsentiert, können Anforderungen anhand der praxisrelevanten Unterscheidungen zwischen der ursprünglich beschriebenen Anforderung aus der Leistungsbeschreibung und der tatsächlich umgesetzten Anforderung bewertet werden. Diese Konstruktionsentscheidung ermöglicht die Zuordnung von Erfahrungswissen eines sicherheitskritischen IT-Projekts zu spezifischen Anforderungen. Abweichungen zwischen der Beschreibung einer Anforderung und der Bewertung der Erfüllung einer Anforderung werden durch die Subklassen der Klasse *SollIstAbweichungAnforderung* ausgedrückt. Mittels nicht-taxonomischer Relationen wird die Verbindung zwischen Beschreibung, Bewertung und Soll-Ist-Abweichung hergestellt. Die Subklassen der Klasse *SollIstAbweichungAnforderung* sind ebenfalls in die Anforderungsbereiche Anforderungen zur Einführung und zum Betrieb, funktionale Anforderungen, nicht-funktionale Anforderungen und technische Anforderungen differenziert. Durch diese Konstruktionsentscheidung können in der Leistungsbeschreibung geforderte Anforderungen mit der durch das sicherheitskritische IT-Projekt bereitgestellten Leistung sowohl als Bewertungswissen als auch als Soll-Ist-Abweichung und durch die sicherheitskritische IT-Projekt-Ontologie ausgedrückt werden. Die nachfolgende Abbildung35 zeigt zusammenfassend die Sachverhalte exemplarisch anhand der funktionalen Anforderung der Datenpflege.

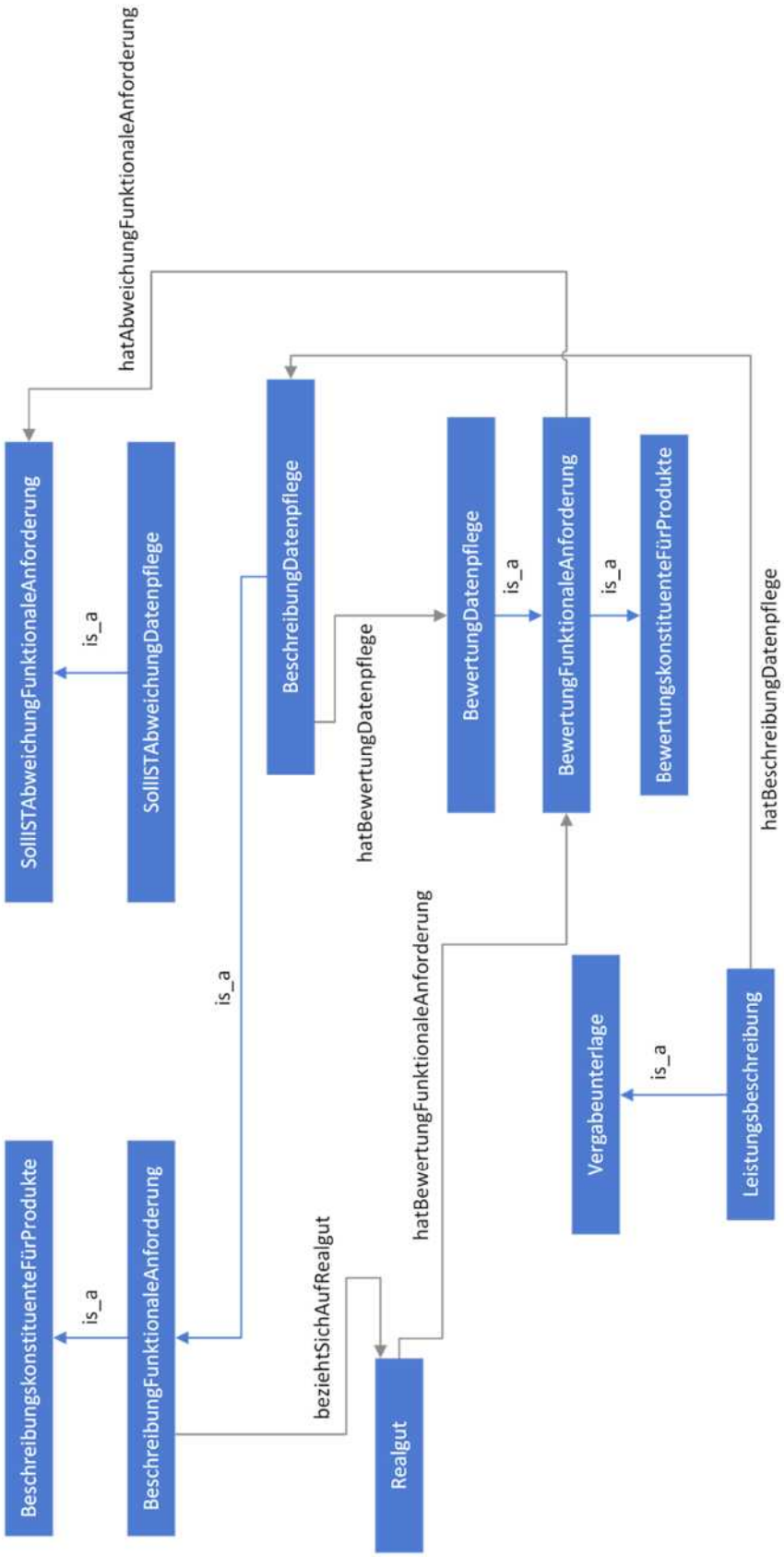


Abbildung 35: Darstellung der Klassenkonstruktion
am Beispiel der funktionalen Anforderung der Datenpflege

Folgende Sachverhalte beschreibt die Abbildung35:

- Die Klasse *BeschreibungskonstituenteFürProdukte* bildet die Oberklasse für die Klasse *BeschreibungFunktionaleAnforderung*, welche wiederum als Oberklasse für die Klasse *BeschreibungDatenpflege* fungiert.
- Die Klasse *Leistungsbeschreibung*, welche eine Subklasse der Klasse *Vergabeunterlage* darstellt, ist mittels einer nicht-taxonomischen Relation *hatBeschreibungDatenpflege* mit der Klasse *hatBeschreibungDatenpflege* verbunden. Diese Konstruktion wird dadurch begründet, dass die beschriebenen Anforderungen – beispielsweise die Datenpflege – aus einer Leistungsbeschreibung entstammen, welche wiederum Teil der Vergabeunterlagen sind.
- Die Klasse *BeschreibungFunktionaleAnforderung* ist mittels einer nicht-taxonomischen Relation *beziehtSichAufRealgut* mit der Klasse *Realgut* verbunden. Die Verwendung dieser nicht-taxonomischen Relation soll den Bezug auf die für die Erfüllung der funktionalen Anforderungen benötigten Realgüter ausdrücken.
- Die Klasse *Realgut* ist mittels einer nicht-taxonomischen Relation *hatBewertungFunktionaleAnforderung* mit der Klasse *BewertungFunktionaleAnforderung* verbunden, welche die Oberklasse für die Klasse *BewertungDatenpflege* darstellt. Diese Konstruktion drückt die Bewertung der verwendeten Realgüter für die Erfüllung der funktionalen Anforderungen zur Datenpflege aus.
- Die Klasse *BewertungFunktionaleAnforderung* ist mittels einer nicht-taxonomischen Relation namens *hatAbweichungFunktionaleAnforderung* mit der Klasse *SollIstAbweichungFunktionaleAnforderung* verbunden. Diese Konstruktion ermöglicht die Darstellung von Abweichungen zwischen der ursprünglich aufgestellten funktionalen Anforderung und der tatsächlichen Anforderungserfüllung. Die Klasse *SollIstAbweichungFunktionaleAnforderung* ist die Oberklasse zur Klasse *SollIstAbweichungDatenpflege*.
- Die Klasse *BeschreibungDatenpflege* ist mittels einer nicht-taxonomischen Relation *hatBewertungDatenpflege* mit der Klasse *BewertungDatenpflege* verknüpft. Diese Konstruktion soll die Bewertung der Erfüllung der beschriebenen funktionalen Anforderungen zur Datenpflege ermöglichen.

Die nachfolgende Tabelle31 erläutert die Subklassen der Klasse *BewertungskonstituenteFürProdukte*.

Klasse	**Beschreibung**	**Subklasse von**
Bewertung Anforderung ZuEinführung UndBetrieb	Die Klasse *BewertungAnforderungZuEinführungUnd Betrieb* ist die Oberklasse für die Bewertungen zu den Anforderungen für die Einführung und den Betrieb von Produkten.	Bewertungs konstituente FürProdukte

Bewertung Funktionale Anforderung	Die Klasse *BewertungFunktionaleAnforderung* ist die Oberklasse für alle Arten von Bewertungen zu den funktionalen Anforderungen.	Bewertungs konstituente FürProdukte
Bewertung NichtFunktionale Anforderung	Die Klasse *BewertungNichtFunktionaleAnforderung* enthält als Oberklasse alle Arten von Bewertungen zu den nicht-funktionalen Anforderungen.	Bewertungs konstituente FürProdukte
Bewertung Technische Anforderung	Die Klasse *BewertungTechnischeAnforderung* ist die Oberklasse für alle Arten von Bewertungen zu den technischen Anforderungen.	Bewertungs konstituente FürProdukte

Tabelle 31: Beschreibung der Subklassen *BewertungskonstituenteFürProdukte*

Die nachfolgende Abbildung36 stellt die Klasse *BewertungskonstituenteFürProjekte* mit den zugehörigen Subklassen dar.

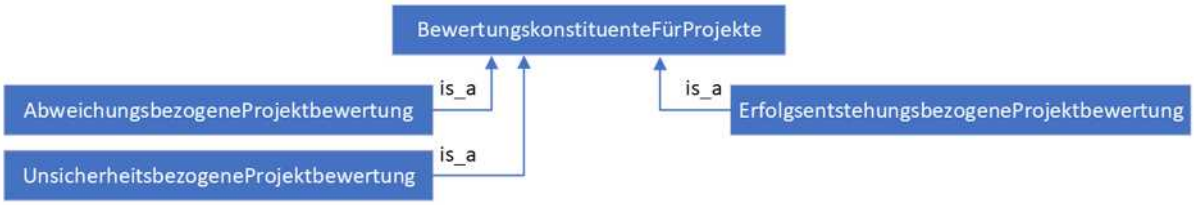


Abbildung 36: Subklassen der Klasse *BewertungskonstituenteFürProjekte*

Die Subklassen der Klassen *AbweichungsbezogeneProjektbewertung, ErfolgsentstehungsbezogeneProjektbewertung* und *UnsicherheitsbezogeneProjektbewertung* stellen sprachliche Ausdrucksmittel bereit, um:

- Risiken eines sicherheitskritischen IT-Projektes auszudrücken (die Klasse *Risikotyp* als Subklasse der Klasse *UnsicherheitsbezogeneProjektbewertung*),
- Abhängigkeitsverhältnisse innerhalb eines sicherheitskritischen IT-Projektes zu verdeutlichen (die Klasse *Abhängigkeitstyp* als Subklasse der Klasse *UnsicherheitsbezogeneProjektbewertung*),
- Beziehungsgeflechte innerhalb eines sicherheitskritischen IT-Projektes darzustellen (die Klasse *Beziehungsgeflecht* als Subklasse der Klasse *UnsicherheitsbezogeneProjektbewertung*),
- Erfolgs- und Misserfolgsfaktoren innerhalb eines sicherheitskritischen IT-Projekte auszudrücken (die Klassen *Erfolgsfaktor* und *Misserfolgsfaktor* als Subklassen der Klasse *ErfolgsentstehungsbezogeneProjektbewertung*),
- Projektqualitätsabweichungen in Form von Soll-Ist-Abweichungen (die Klasse *SollIstAbweichungAnforderung* als Subklasse der Klasse *AbweichungsbezogeneProjektbewertung*) zu repräsentieren.

Die nachfolgende Tabelle 32 erläutert die Subklassen der Klasse *BewertungskonstituenteFür-Projekte*.

Klasse	**Beschreibung**	**Subklasse von**
Abweichungsbezogene Projektbewertung	Die Klasse *Abweichungsbezogene Projektbewertung* ist die Oberklasse für alle Arten von abweichungsbezogenen Projektbewertungen.	Bewertungs konstituente FürProjekte
Unsicherheitsbezogene Projektbewertung	Die Klasse *Unsicherheitsbezogene Projektbewertung* ist die Oberklasse für alle Arten von unsicherheitsbezogenen Projektbewertungen.	Bewertungs konstituente FürProjekte
Erfolgsentstehungsbezo-gene Projektbewertung	Die Klasse *Erfolgsentstehungsbezogene Projektbewertung* ist die Oberklasse für alle Arten von erfolgsentstehungs-bezogenen Projektbewertungen.	Bewertungs konstituente FürProjekte

Tabelle 32: Beschreibung der Subklassen *BewertungskonstituenteFürProjekte*

Die nachfolgende Abbildung37 zeigt die Klasse *ImmateriellesRealgut* mit den zugehörigen Subklassen.

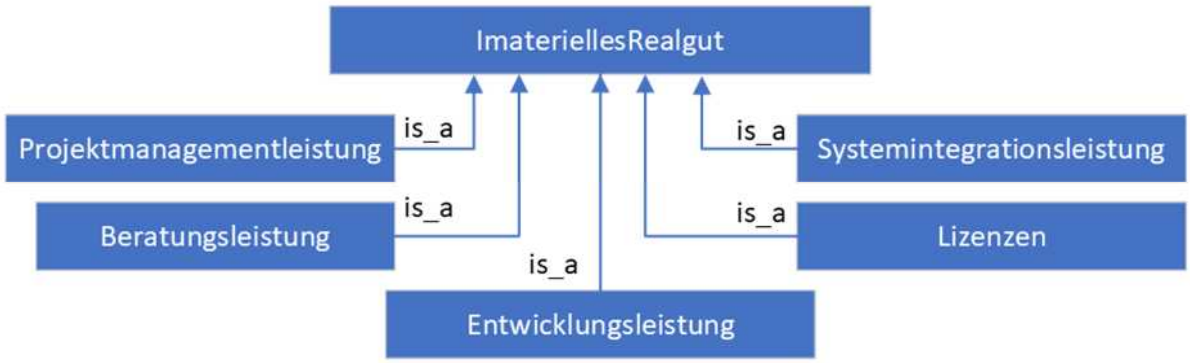


Abbildung 37: Subklassen der Klasse *ImmateriellesRealgut*

Die Differenzierung der Klasse *ImmateriellesRealgut* in die weiteren Subklassen basiert im Wesentlichen auf der Grundlage der verwendeten Leistungsbeschreibungen. In den zugrunde gelegten Leistungsbeschreibungen werden verschiedene immaterielle Realgüter (sowohl im Sinne von Produktionsfaktoren als auch von Produkten) gefordert, die in die nachfolgenden Bereiche differenziert werden:

- Projektmanagementleistung
- Beratungsleistung
- Entwicklungsleistung
- Lizenzen
- Systemintegrationsleistung

Diese Leistungsarten stellen die Subklassen der Klasse *ImmateriellesRealgut* dar und werden in der nachfolgenden Tabelle33 erläutert.

Klasse	**Beschreibung**	**Subklasse von**
Projektmanagement leistung	Die Klasse *Projektmanagementleistung* ist die Oberklasse für alle Arten von Projektmanagementleistungen.	Immaterielles Realgut
Beratungsleistung	Die Klasse *Beratungsleistung* ist die Oberklasse für alle Arten von Beratungsleistungen.	Immaterielles Realgut
Entwicklung leistung	Die Klasse *Entwicklungsleistung* enthält als Oberklasse alle Arten von Entwicklungs-leistungen.	Immaterielles Realgut
Lizenzen	Die Klasse *Lizenzen* ist die Oberklasse für alle Arten von Lizenzen.	Immaterielles Realgut
Systemintegrations leistung	Die Klasse *Systemintegrationsleistung* ist die Oberklasse für alle Arten von Systemintegrationsleistungen.	Immaterielles Realgut

Tabelle 33: Beschreibung der Subklassen der Klasse *ImmateriellesRealgut*

Die nachfolgende Abbildung 38 stellt die Klasse *MateriellesRealgut* mit den zugehörigen Subklassen dar.

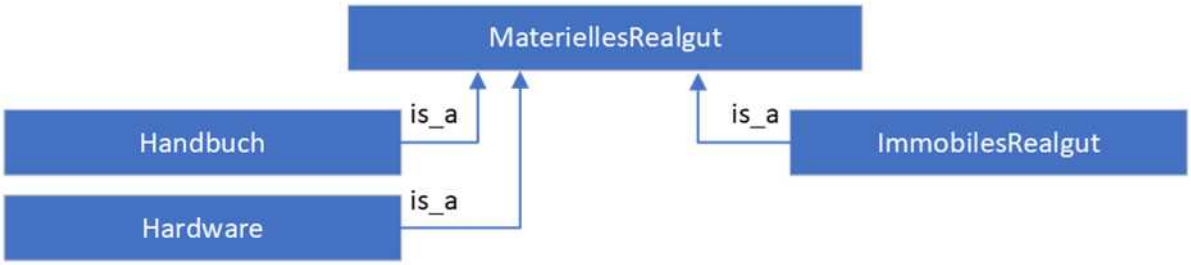


Abbildung 38: Subklassen der Klasse *MateriellesRealgut*

Analog zur Differenzierung der Klasse *ImmateriellesRealgut* erfolgt die Unterteilung der Klasse *MateriellesRealgut* primär auf Basis der in den Leistungsbeschreibungen geforderten materiellen Realgüter. In den zugrunde gelegten Leistungsbeschreibungen werden verschiedene materielle Realgüter gefordert, die in die nachfolgenden Bereiche unterteilt werden:

- Handbuch
- Hardware
- immobile Realgüter

Diese Bereiche stellen die Subklassen der Klasse *MateriellesRealgut* dar und werden in der nachfolgenden Tabelle 34 erläutert.

Klasse	**Beschreibung**	**Subklasse von**
Handbuch	Die Klasse *Handbuch* ist die Oberklasse für alle Arten von Handbüchern.	Materielles Realgut
Hardware	Die Klasse *Hardware* ist die Oberklasse für alle Arten von Hardware.	Materielles Realgut
ImmobilesRealgut	Die Klasse *ImmobilesRealgut* ist die Oberklasse aller Arten von immobilen Realgütern.	Materielles Realgut

Tabelle 34: Beschreibung der Subklassen der Klasse *MateriellesRealgut*

Die nachfolgende Abbildung39 zeigt die Klasse *BeschreibungNichtFunktionaleAnforderung* mit den zugehörigen Subklassen.

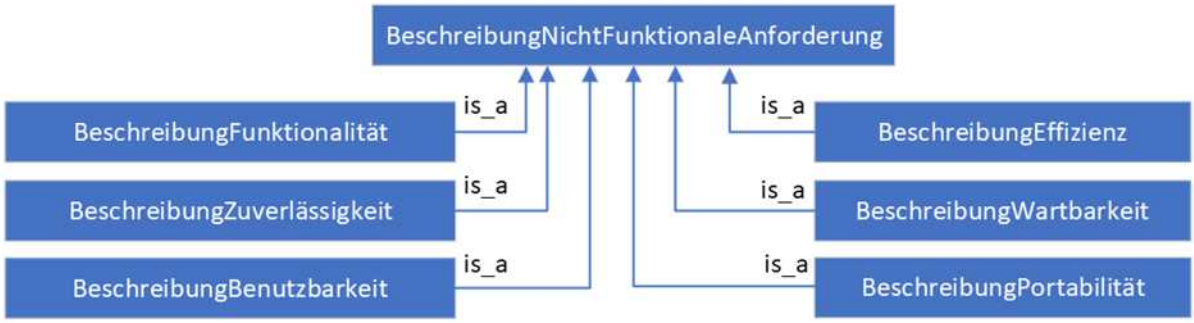


Abbildung 39: Subklassen der Klasse *BeschreibungNichtFunktionaleAnforderung*

Nicht-funktionale Anforderungen werden gemäß der ISO-Norm 9126 in verschiedene Merkmalskategorien unterschieden. Die zugrunde gelegten Leistungsbeschreibungen lehnen sich an diese Norm an; vgl. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2000), S. 7-13. Dies begründet die Differenzierung der Subklassen der Klasse *BeschreibungNichtFunktionale-Anforderung* in die Merkmale, die in der nachfolgenden Tabelle 35 erläutert werden.

Merkmal	**Definition**
Funktionalität	Das Merkmal Funktionalität beschreibt, dass die erwartete Funktionalität und der beabsichtigte Nutzen erfüllt werden.
Zuverlässigkeit	Das Merkmal Zuverlässigkeit spezifiziert die Fähigkeit, dass das Produkt ohne Fehler arbeitet und das festgelegte Leistungsniveau bei normalen Einsatzbedingungen beibehält.
Benutzbarkeit	Das Merkmal Benutzbarkeit spezifiziert den benötigen Aufwand, um die Nutzung des Produkts zu erlernen.
Effizienz	Das Merkmal Effizienz beschreibt die Fähigkeit eines Produkts, eine angemessene Leistung unter Berücksichtigung von Ressourcen wie Zeit, Speicherplatz oder Energieverbrauch zu erzielen und die Anforderungen an das Zeitverhalten einzuhalten
Wartbarkeit	Das Merkmal Wartbarkeit spezifiziert die Fähigkeit des Produkts, änderungsfähig zu sein, um Korrekturen, Verbesserungen und Modifikationen vornehmen zu können.
Portabilität	Das Merkmal Portabilität spezifiziert die Fähigkeit des Produkts, von einer Umgebung in eine andere übertragen werden zu können.

Tabelle 35: Merkmale der ISO-Norm 9126

Die nachfolgende Tabelle36 erläutert auf Basis der in Tabelle 35 definierten Merkmale die Subklassen der Klasse *BeschreibungNichtFunktionaleAnforderung*.

Klasse	**Beschreibung**	**Subklasse von**
Beschreibung Funktionalität	Die Klasse *BeschreibungFunktionalität* ist die Oberklasse für alle Arten der Beschreibungen der nicht-funktionalen Anforderungen, die sich auf den Merkmalsbereich der Funktionalität beziehen.	Beschreibung NichtFunktionale Anforderung

Beschreibung Zuverlässigkeit	Die Klasse *BeschreibungZuverlässigkeit* ist die Oberklasse für alle Arten der Beschreibungen der nicht-funktionalen Anforderungen, die sich auf den Merkmalsbereich der Zuverlässigkeit beziehen.	Beschreibung NichtFunktionale Anforderung
Beschreibung Benutzbarkeit	Die Klasse *BeschreibungBenutzbarkeit* enthält als Oberklasse für alle Arten der Beschreibungen der nicht-funktionalen Anforderungen, die sich auf den Merkmalsbereich der Benutzbarkeit beziehen.	Beschreibung NichtFunktionale Anforderung
Beschreibung Effizienz	Die Klasse *BeschreibungEffizienz* ist die Oberklasse für alle Arten der Beschreibungen der nicht-funktionalen Anforderungen, die sich auf den Merkmalsbereich der Effizienz beziehen.	Beschreibung NichtFunktionale Anforderung
Beschreibung Wartbarkeit	Die Klasse *BeschreibungWartbarkeit* ist die Oberklasse für alle Arten der Beschreibungen der nicht-funktionalen Anforderungen, die sich auf den Merkmalsbereich der Wartbarkeit beziehen.	Beschreibung NichtFunktionale Anforderung
Beschreibung Portabilität	Die Klasse *BeschreibungPortabilität* ist die Oberklasse für alle Arten der Beschreibungen der nicht-funktionalen Anforderungen, die sich auf den Merkmalsbereich der Portabilität beziehen.	Beschreibung NichtFunktionale Anforderung

Tabelle 36: Beschreibung der Subklassen der Klasse *BeschreibungNichtFunktionaleAnforderung*

Die Klasse Risikotyp dient als sprachliches Ausdrucksmittel zur Beschreibung von Risiken in sicherheitskritischen IT-Projekten. Sie wird in die Subklassen *MittelbaresRisiko*, *UnmittelbaresRisiko* und *Unsicherheit* ausdifferenziert. Diese Unterscheidung resultiert aus der Tatsache, dass mittelbare Risiken durch äußere Einflüsse entstehen und bedingt beeinflussbar sind, während unmittelbare Risiken projektinterne Ursachen haben und in der Regel beeinflussbar sind. Unsicherheiten können benannt, jedoch noch nicht bewertet werden.

Die nachfolgende Abbildung40 veranschaulicht exemplarisch die taxonomischen und nicht-taxonomischen Relationen der Subklassen der Klasse *Risikotyp* im Bereich der funktionalen Anforderungen. Dadurch wird deutlich, welche konzeptuellen Entscheidungen der sicherheitskritischen IT-Projekt-Ontologie zugrunde liegen, um Risiken, die sich bei jeder Anforderung ergeben können, sprachlich zu strukturieren.

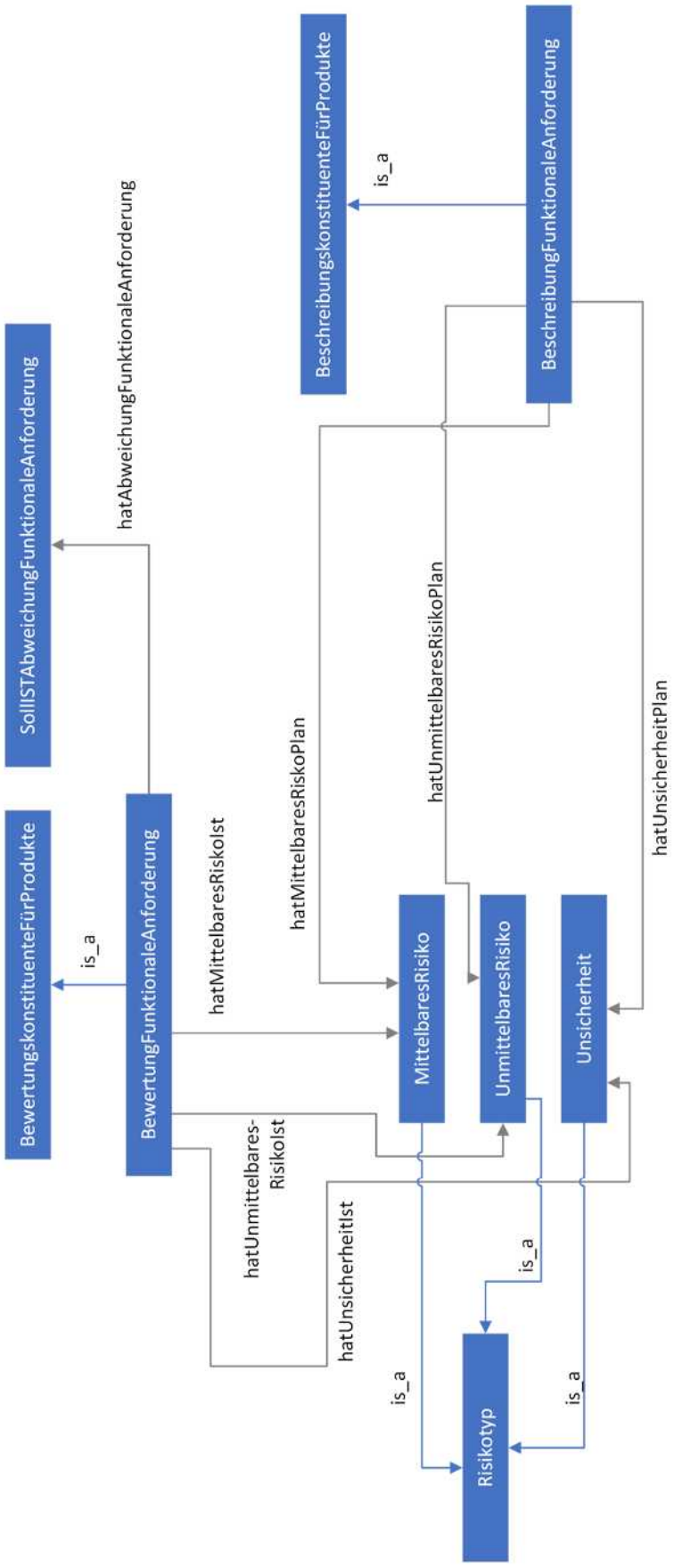


Abbildung 40: Strukturierung der Risiken am Beispiel der funktionalen Anforderungen

Die Abbildung40 stellt folgende Sachverhalte dar:

- Die Klasse *BeschreibungFunktionaleAnforderung* ist mit einer nicht-taxonomischen Relation (*hatMittelbaresRisikoPlan*) mit der Klasse *MittelbaresRisiko* verbunden. Diese Konstruktion ermöglicht es, die beschriebenen funktionalen Anforderungen den geplanten mittelbaren Risiken zuzuordnen.
- Die Klasse *BeschreibungFunktionaleAnforderung* ist mit einer nicht-taxonomischen Relation (*hatUnmittelbaresRisikoPlan*) mit der Klasse *UnmittelbaresRisiko* verbunden. Diese Konstruktion ermöglicht es, die beschriebenen funktionalen Anforderungen den geplanten unmittelbaren Risiken zuzuordnen.
- Die Klasse *BeschreibungFunktionaleAnforderung* ist mit einer nicht-taxonomischen Relation (*hatUnmittelbareUnsicherheitPlan*) mit der Klasse *Unsicherheit* verbunden. Diese Konstruktion ermöglicht es, die beschriebenen funktionalen Anforderungen den geplanten Unsicherheiten zuzuordnen.
- Die Klasse *BewertungFunktionaleAnforderung* ist durch eine nicht-taxonomische Relation (*hatmittelbaresRisikoIst*) mit der Klasse *mittelbaresRisiko* verknüpft. Diese Konstruktion ermöglicht es, die funktionalen Anforderungen den tatsächlich eingetretenen mittelbaren Risiken zuzuordnen.
- Die Klasse *BewertungFunktionaleAnforderung* ist durch eine nicht-taxonomische Relation (*hatUnmittelbaresRisikoIst*) mit der Klasse *UnmittelbaresRisiko* verknüpft. Diese Konstruktion ermöglicht es, die funktionalen Anforderungen den tatsächlich eingetretenen unmittelbaren Risiken zuzuordnen.
- Die Klasse *BewertungFunktionaleAnforderung* ist durch eine nicht-taxonomische Relation (*hatUnsicherheitIst*) mit der Klasse *Unsicherheit* verknüpft. Diese Konstruktion ermöglicht es, die funktionalen Anforderungen den tatsächlich eingetretenen Unsicherheiten zuzuordnen.

Die nachfolgende Tabelle37 erläutert die Subklassen der Klasse *Risikotyp*:

Klasse	**Beschreibung**	**Subklasse von**
UnmittelbaresRisiko	Die Klasse *UnmittelbaresRisiko* ist die Oberklasse für alle Arten von unmittelbaren Risiken.	Risikotyp
MittelbaresRisiko	Die Klasse M*ittelbaresRisiko* ist die Oberklasse für alle Arten von mittelbaren Risiken.	Risikotyp
Unsicherheit	Die Klasse *Unsicherheit* ist die Oberklasse für alle Arten von Unsicherheiten	Risikotyp

Tabelle 37: Beschreibung der Subklassen der Klasse *Risikotyp*

3.2.3.5 Konstruktion der nicht-taxonomischen Relationen

Nachfolgend wird auf die Konstruktion von Eigenschaften eingegangen. Hinsichtlich der Konstruktion der Eigenschaften muss zwischen nicht-taxonomischen Relationen und Attributen unterschieden werden. Zunächst wird nur die Konstruktion von nicht-taxonomischen Relationen (auch kurz als „Relationen“ bezeichnet) betrachtet. Die Konstruktion von Attributen wird später in Kapitel 3.2.3.6 erläutert.

Die nicht-taxonomischen Relationen werden in den nachfolgenden Ausführungen in einer tabellarischen Darstellung angegeben, die wie folgt aufgebaut ist:

Relationsbezeichnung	**Domain**	**Range**

Tabelle 38: Tabellenstruktur für die Erläuterung der nicht-taxonomischen Relationen

Eine nicht-taxonomische Relation hat eine Relationsbezeichnung und wird mit der zugehörigen Domain (Vorbereich) sowie Range (Nachbereich) spezifiziert. Die nachfolgende Abbildung 41 zeigt die relevanten Komponenten einer nicht-taxonomischen Relation.

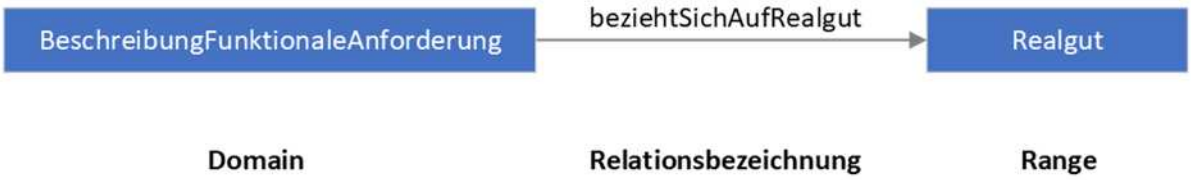


Abbildung 41: Komponenten einer nicht-taxonomischen Relation

Die Abbildung41 stellt exemplarisch die nicht-taxonomische Relation *beziehtSichAufRealgut* dar, die als Domain die Klasse *BeschreibungFunktionaleAnforderung* und als Range die Klasse *Realgut* zugewiesen hat. Mittels der logischen Ausdrücke AND und OR können sowohl in der Domäne als auch in der Range weitere Klassen hinzugefügt werden. In dem genannten Beispiel verbindet die Relation *beziehtSichAufRealgut* Instanzen der Klasse *BeschreibungFunktionale-Anforderung* (Domain) mit Instanzen der Klasse *Realgut* (Range).

In der vorliegenden PM-Domänen-Ontologie sowie für die PRINCE2- und Risikomanagement-Ontologie sind bereits einige nicht-taxonomische Relationen vordefiniert. Allerdings reichen diese aufgrund der spezifischen Anforderungen von sicherheitskritischen IT-Projekten nicht aus. Beispiele für eine spezifische Anforderung an die sicherheitskritische IT-Projekt-Ontologie sind die notwendigen sprachlichen Ausdrucksmittel, die in den zugrunde gelegten Leistungsbeschreibungen verwendet werden. Durch die Einführung zusätzlicher Klassen in der sicherheitskritischen IT-Projekte-Ontologie müssen auch nicht-taxonomische Relationen konstruiert werden, um die Ausdrucksfähigkeit der sicherheitskritischen IT-Projekte-Ontologie zu erhöhen.

In den nachfolgenden Erläuterungen werden ausschließlich die neu konstruierten nicht-taxonomischen Relationen behandelt. Auf die vordefinierten Relationen wird nicht weiter eingegangen.

Nachfolgend werden die Konstruktionsschritte erläutert, die erforderlich sind, um eine nicht-taxonomische Relationen in Protégé anzulegen. Das Anlegen einer nicht-taxonomischen Relation erfolgt unter dem Punkt „Object Properties“, wie die nachfolgende Abbildung42 zeigt.

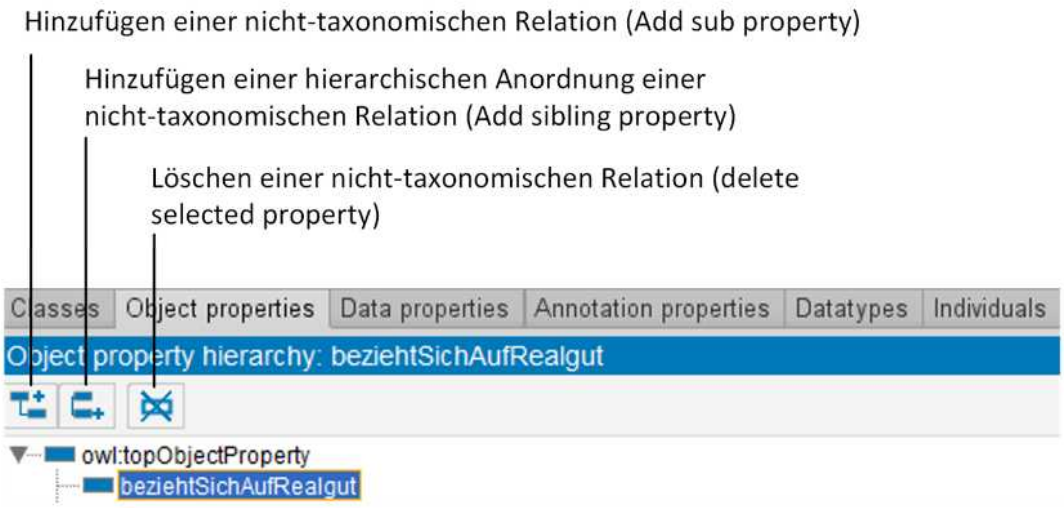


Abbildung 42: Object Properties in Protégé

Protégé bietet zwei Möglichkeiten an, um nicht-taxonomische Relationen zu konstruieren. In diesem Beitrag wird die Möglichkeit „Add Sub Property“ genutzt, um eine nicht-taxonomische Relation zu konstruieren. Es besteht jedoch auch die Möglichkeit, eine nicht-taxonomische Relation zu konstruieren, die in einem Subsumtionsverhältnis zu einer anderen nicht-taxonomischen Relation steht. Bei der Konstruktion als Subsumtionsverhältnis (dies betrifft die Option „Add Sibling property“ in der Abbildung42) erbt die untergeordnete nicht-taxonomische Relation die Eigenschaft der übergeordneten nicht-taxonomischen Relation; vgl. DEBELLIS (2021), S. 22. In der sicherheitskritischen IT-Projekt-Ontologie wird keine nicht-taxonomische Relation konstruiert, die ein Subsumtionsverhältnis besitzt. Diese Konstruktionsentscheidung wird dahingehend begründet, dass es sich bei der Spezifizierung der nicht-taxonomischen Relationen als nicht zweckmäßig herausgestellt hat, ein Subsumtionsverhältnis abzubilden. Es existieren zwar nicht-taxonomische Relationen, bei denen sich die Spezifizierung eines Subsumtionsverhältnisses angeboten hätte, jedoch ist für die Verwendung in einem CBR-System die Verwendung eines Subsumtionsverhältnisses unerheblich.

Sobald die nicht-taxonomischen Relationen konstruiert wurden, werden alle nicht-taxonomischen Relationen der von Protégé als Standard vorgegebenen Relation *owl:topObjectProperty* untergeordnet, wie die Abbildung42 exemplarisch für die nicht-taxonomische Relation *beziehtSichAufRealgut* zeigt.

Die nachfolgende Abbildung43 stellt mittels des Class Expression Editors die Konstruktion der Klassen für die Domäne der nicht-taxonomischen Relation *beziehtSichAufRealgut* dar. Mittels des Class Expression Editors kann die Zuweisung der Klasse für den Bereich „Range“ in analoger Weise zu der Zuweisung der Domäne erfolgen.

In dem genannten Beispiel werden dieser nicht-taxonomischen Relation in der Domäne mehrere Klassen zugewiesen, die mit dem logischen Ausdruck OR verbunden werden. Dies bedeutet, dass jede der genannten Klassen zulässig ist, wodurch die Vereinigungsmenge ihrer Instanzen zur Instanziierung im Vorbereich der Relation zur Verfügung steht.

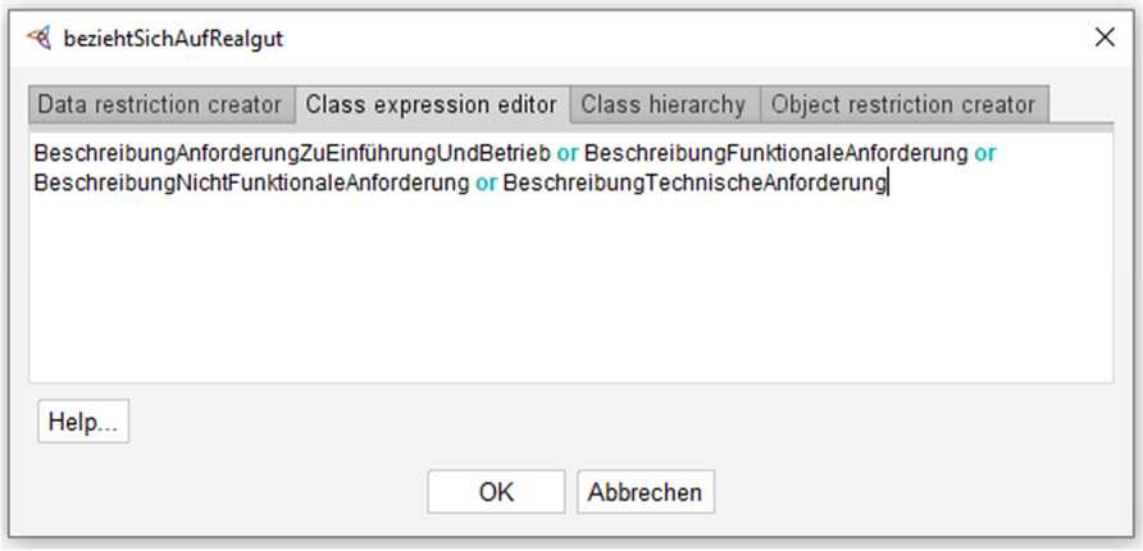


Abbildung 43: Class Expression Editor
mit der Zuweisung einer Domäne zu einer nicht-taxonomischen Relation

Das Ergebnis der Zuweisung kann im Bereich „Description: *beziehtSichAufRealgut*“, wie die nachfolgende Abbildung 444 illustriert, eingesehen werden.

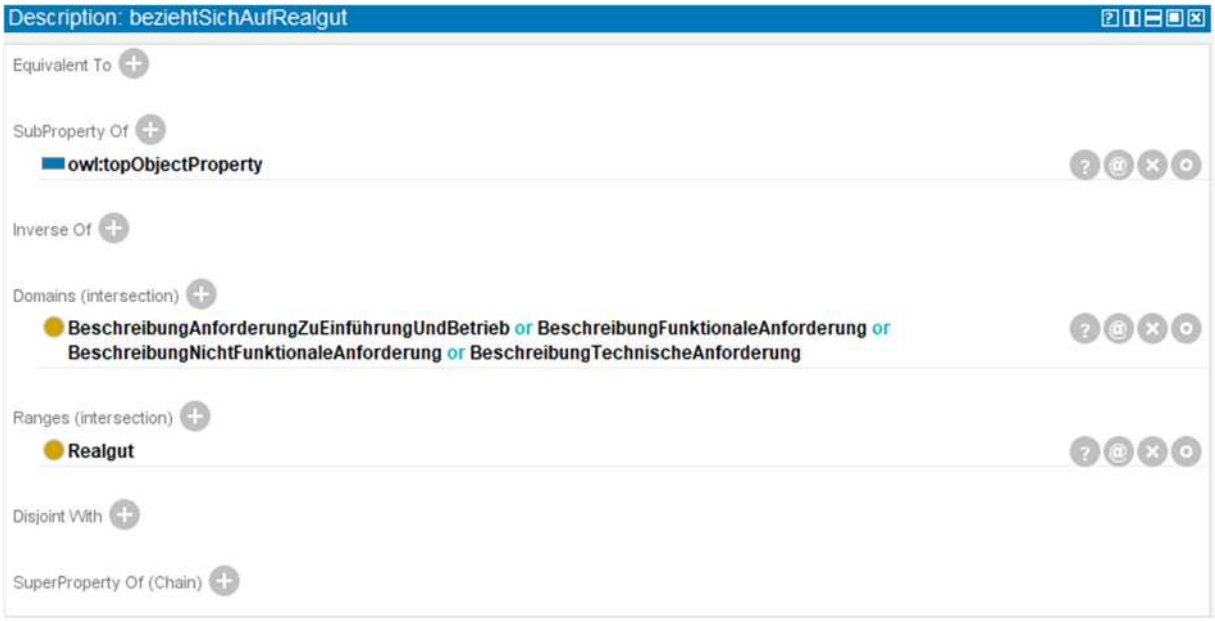


Abbildung 44: Beschreibung einer nicht-taxonomischen Relation in Protégé

Die Ausdruckskraft von nicht-taxonomischen Relationen kann durch weitere Charakteristika in Protégé erweitert werden. Dazu bietet Protégé für nicht-taxonomische Relationen folgende Charakteristika an: funktional, invers funktional, transitiv, symmetrisch, asymmetrisch, reflexiv und irreflexiv. Die Auswahl der Charakteristika erfolgt in einem eigenen Auswahlfenster in Protégé, welches als Auswahlfenster die Bezeichnung „Characteristics“ trägt. Für die Konstruktion der sicherheitskritischen IT-Projekt-Ontologie wurde auf die Verwendung der Charakteristika verzichtet. Ähnlich zur Verwendung der Subsumtion von nicht-taxonomischen Relationen existieren zwar nicht-taxonomische Relationen, bei denen sich die Verwendung der vorgenannten Charakteristika möglicherweise angeboten hätte, die jedoch für die Verwendung in einem CBR-System unerheblich sind. Die Nutzung von Charakteristika wird in einigen Quellen auch kontrovers diskutiert. So empfiehlt DEBELLIS (2021), S. 26, die bedachte Nutzung der Charakteristika „reflexiv“ aufgrund der Auswirkung auf die Schlussfolgerungskomponente (Reasoner). Für eine weitergehende Erläuterung der Charakteristika wird auf DEBELLIS (2021), S. 24-26, verwiesen.

Im Nachfolgenden werden exemplarische nicht-taxonomische Relationen in Verbindung mit zentralen Klassen der sicherheitskritischen IT-Projekt-Ontologie hinsichtlich ihrer Konstruktion und ihrer Interdependenzen erläutert. Siehe dazu die nachfolgende Abbildung45.

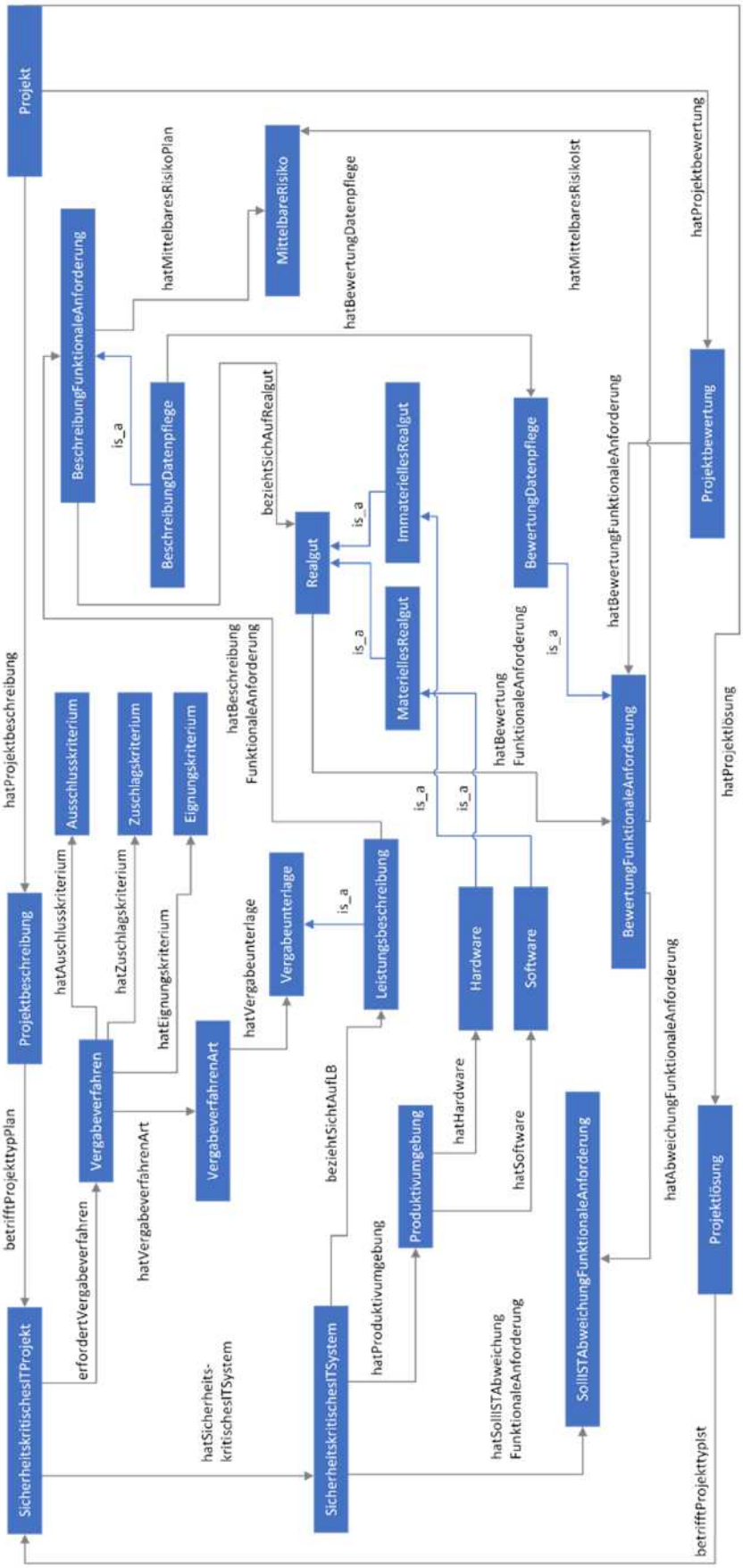


Abbildung 45: Exemplarischer Auszug aus der sicherheitskritischen IT-Projekt-Ontologie

Die Abbildung 45 stellt hinsichtlich der nicht-taxonomischen Relationen folgende Sachverhalte dar:

- Die Klasse *Projektbeschreibung* und die Klasse *SicherheitskritischesITProjekt* werden mittels einer nicht-taxonomischen Relation (*betrifftProjekttypPlan*) verbunden. Die Klasse *Projektbeschreibung* ist eine zentrale Klasse für das CBR-System, und durch die Konstruktion der nicht-taxonomischen Relation (*betrifftProjekttypPlan*) werden die sprachlichen Ausdrucksmittel bereitgestellt, um dann ausgehend von der Klasse Projekttyp die geplanten Projektcharakteristika zu beschreiben und diese der Projektbeschreibung zuzuordnen.
- Die Klasse *SicherheitskritischesITProjekt* und die Klasse *Vergabeverfahren* werden mittels einer nicht-taxonomischen Relation (*erfordertVergabeverfahren*) verbunden. Durch diese Konstruktionsentscheidung werden sprachliche Ausdrucksmittel bereitgestellt, um einem Vergabeverfahren ein sicherheitskritisches IT-Projekt zu zuweisen.
- Die Klasse *Vergabeverfahren* ist durch die nicht-taxonomischen Relationen *hatAusschlusskriterium*, *hatZuschlagskriterium* und *hatEignungskriterium* mit den Klassen *Ausschlusskriterium*, *Zuschlagskriterium* und *Eignungskriterium* verbunden. Diese Konstruktion ergibt sich aus der Tatsache, dass ein Vergabeverfahren Ausschlusskriterien, Zuschlagskriterien und Eignungskriterien besitzt, um potenzielle Anbieter anhand dieser Kriterien zu bewerten.
- Die Klasse *Vergabeverfahren* ist mittels einer nicht-taxonomischen Relation, und zwar *hatVergabeverfahrenArt*, mit der Klasse *VergabeverfahrenArt* verbunden. Diese Konstruktion ermöglicht es, einem Vergabeverfahren eine spezifische Vergabeverfahrensart zuzuweisen.
- Die Klasse *VergabeverfahrensArt* ist über eine nicht-taxonomische Relation *hatVergabeunterlage* mit der Klasse *Vergabeunterlage* verbunden. Diese Konstruktion ermöglicht es, die Unterlagen, die im Kontext eines spezifischen Vergabeverfahrens bereitgestellt werden, wie zum Beispiel eine Leistungsbeschreibung (ausgedrückt als Klasse *Leistungsbeschreibung*) in der sicherheitskritischen IT-Projekte-Ontologie auszudrücken.
- Die Klasse *SicherheitskritischesITProjekt* ist durch die nicht-taxonomische Relation *hatSicherheitskritischesITSystem* mit der Klasse *SicherheitskritischesITSystem* verknüpft. Diese Konstruktionsentscheidung erfolgt im Einklang mit der in diesem Beitrag vorgestellten Definition eines sicherheitskritischen IT-Projekts, wonach ein sicherheitskritisches IT-Projekt als Liefergegenstand ein sicherheitskritisches IT-System hat.
- Die Klasse *SicherheitskritischesITSystem* ist über die nicht-taxonomische Relation *beziehtSichAufLB* mit der Klasse *Leistungsbeschreibung* verbunden. Durch diese Konstruktionsentscheidung soll ausgedrückt werden, auf welcher Leistungsbeschreibung sich das sicherheitskritische IT-System bezieht.

- Die Klasse *Leistungsbeschreibung* ist mittels der nicht-taxonomischen Relation *hatBeschreibungFunktionaleAnforderung* mit der Klasse *BeschreibungFunktionaleAnforderung* verknüpft. Ziel dieser Konstruktion ist es, die in einer Leistungsbeschreibung formulierten konkreten funktionalen Anforderungen zu referenzieren. Im Rahmen öffentlicher Vergabeverfahren werden Anforderungen, wie bereits in diesem Beitrag erläutert, in Leistungsbeschreibungen formuliert, weshalb diese Konstruktionsentscheidung dazu dient, diesen Sachverhalt für den Bereich der funktionalen Anforderungen auszudrücken.
- Die Klasse *SicherheitskritischesITSystem* ist durch die nicht-taxonomische Relation *hatProduktivumgebung* mit der Klasse *Produktivumgebung* verbunden. Ziel dieser Konstruktion ist es, die Produktivumgebung eines sicherheitskritischen IT-Systems auszudrücken. Die Klasse *Produktivumgebung* ist durch die zwei nicht-taxonomischen Relationen *hatHardware* sowie *hatSoftware* mit den Klassen *Hardware* bzw. *Software* verbunden. Beide Klassen sind Subklassen der Klasse *Realgut*. Auf diese Weise soll ausgedrückt werden, aus welchen materiellen und immateriellen Realgütern, in diesem Fall Software und Hardware, die Produktivumgebung eines sicherheitskritischen IT-Systems besteht.
- Die Klasse Projektlösung ist mittels der nicht-taxonomischen Relation *betrifftProjekttypIst* mit der Klasse *SicherheitskritischesITProjekt* verbunden. Diese Konstruktion ermöglicht es, auf alle relevanten Klassen und Eigenschaften zuzugreifen, um die tatsächliche Projektlösung eines sicherheitskritischen IT-Projektes auszudrücken. Zusätzlich werden mittels der nicht-taxonomischen Relation *hatAbweichungFunktionaleAnforderung*, welche die Klasse *BewertungFunktionaleAnforderung* mit der Klasse *SollISTAbweichungFunktionaleAbweichung* verbindet, die Ausdrucksmittel bereitgestellt, um neben der realisierten Projektlösung auch die Abweichung von der geplanten Lösung auszudrücken.

Eine vollständige Auflistung aller nicht-taxonomischen Relationen der sicherheitskritischen IT-Projekt-Ontologie findet sich in SETHUPATHY (2024), S. 243-257.

Einschränkend ist zu erwähnen, dass N-stellige Relationen mit $N>2$ in rechnergestützten Wissensmanagementsystemen in der Regel nicht unterstützt werden. Stattdessen sind nur zweistellige nicht-taxonomische Relationen üblich. Diese Einschränkung führt dazu, dass beispielsweise keine nicht-taxonomische Relation spezifiziert werden kann, die aussagt, dass für ein spezifisches sicherheitskritisches IT-System *X* eine Software *Y* notwendig ist, die nur mit der Hardware *Z* funktioniert (ternäre Relation). Um diesen Sachverhalt darzustellen, werden zwei nicht-taxonomische Relationen benötigt, wobei die Klasse *Software* im Vor- oder Nachbereich der beiden nicht-taxonomischen Relationen (*hatSoftware*, *benötigtHardware*) vorkommen.

3.2.3.6 Konstruktion der Attribute

Nachfolgend wird auf die Konstruktion der Attribute eingegangen. Die Attribute werden in einer tabellarischen Darstellung angegeben, welche wie folgt aufgebaut ist:

Domain	**Attributbezeichnung**	**Range**

Tabelle 39: Tabellenstruktur für die Erläuterung der Attribute

Ein Attribut hat eine Attributbezeichnung und wird mit seiner Domain (Vorbereich) sowie seiner Range (Nachbereich) definiert. Im Gegensatz zu nicht-taxonomischen Relationen bezieht sich die Range nicht auf eine Klasse, sondern auf einen Datentyp. Ein Datentyp definiert, wie viele Bytes eine Variable ab ihrer Adresse im Hauptspeicher eines Rechnersystems belegt sowie wie das Bitmuster dieser Bytes interpretiert wird. In diesem Beitrag werden ausschließlich primitive Datentypen verwendet, da diese sowohl vom CBR-Tool jCORA unterstützt als auch in der einschlägigen Fachliteratur empfohlen werden. Die Entscheidung, sich auf primitive Datentypen zu beschränken, wurde getroffen, um eine höhere Kompatibilität mit anderen Systemen und eine bessere Austauschbarkeit der Daten zu gewährleisten.

Die nachfolgende Abbildung46 zeigt die relevanten Komponenten eines Attributs in exemplarischer Weise anhand der Attribute der Klasse *SicherheitskritischesITProjekt*.

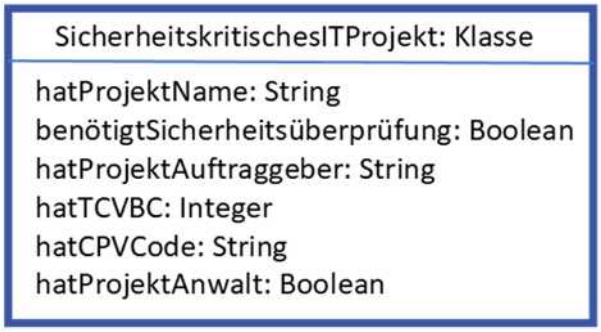


Abbildung 46: Exemplarische Darstellung der Attribute der Klasse *SicherheitskritischesITProjekt*

In der zugrunde gelegten PM-Domänen-Ontologie und der PRINCE2- und Risikomanagement-Ontologie sind bereits einige Attribute vorkonstruiert. Die bereits vorliegenden Attribute sind jedoch für die sicherheitskritische IT-Projekt-Ontologie u. a. aufgrund der zusätzlich konstruierten Klassen nicht ausreichend.

Die Konstruktion der Attribute erfolgt in Protégé im Bereich „Data Properties“, wie die nachfolgende Abbildung47 zeigt.

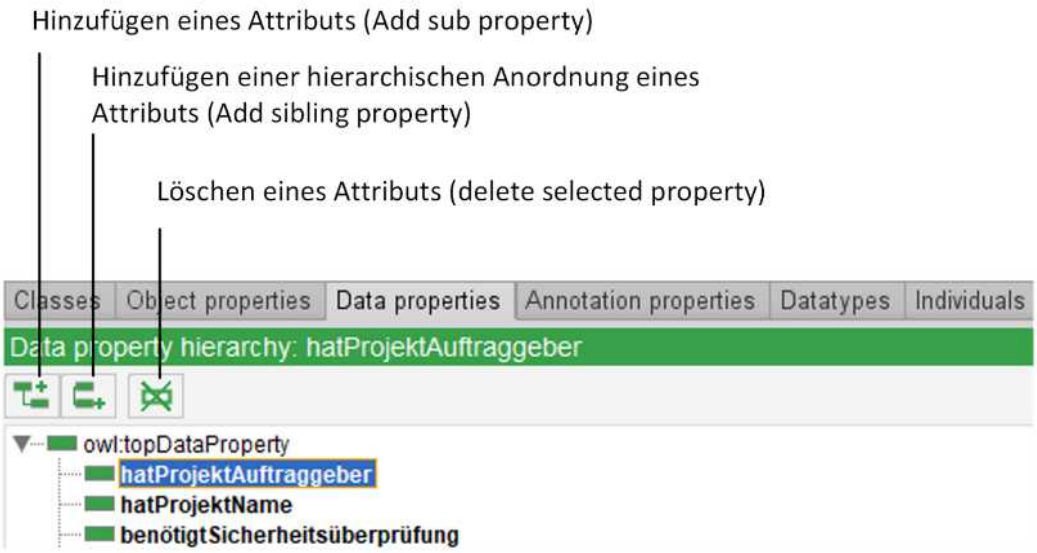


Abbildung 47: Data properties in Protégé

Protégé bietet zwei Möglichkeiten an, um Attribute zu konstruieren. In diesem Beitrag wird die Möglichkeit „Add Sub Property“ genutzt, um Attribute zu konstruieren. Es besteht jedoch auch die Möglichkeit, Attribute zu konstruieren, die in einem Subsumtionsverhältnis zu anderen Attributen stehen. Die Konstruktion erfolgt durch die Option „Add Sibling property“.

Sobald die Attribute konstruiert wurden, werden alle Attribute dem standardmäßig von Protégé vorgegebenen Attribut *owl:topDataProperty* untergeordnet, wie die Abbildung47 exemplarisch für die Attriute *hatProjektName*, *benötigtSicherheitsüberprüfung* und *hatProjektAuftraggeber* zeigt.

Die nachfolgende Abbildung48 stellt mittels des Class Expression Editors die Konstruktion des Attributs *benötigtSicherheitsüberprüfung* dar, dem als Domäne (Vorbereich) die Klasse *SicherheitskritischesITProjekt* zugeordnet ist. In der Range (Nachbereich) des Attributs wird der Datentyp Boolean zugewiesen, wie die übernächste Abbildung49 zeigt.

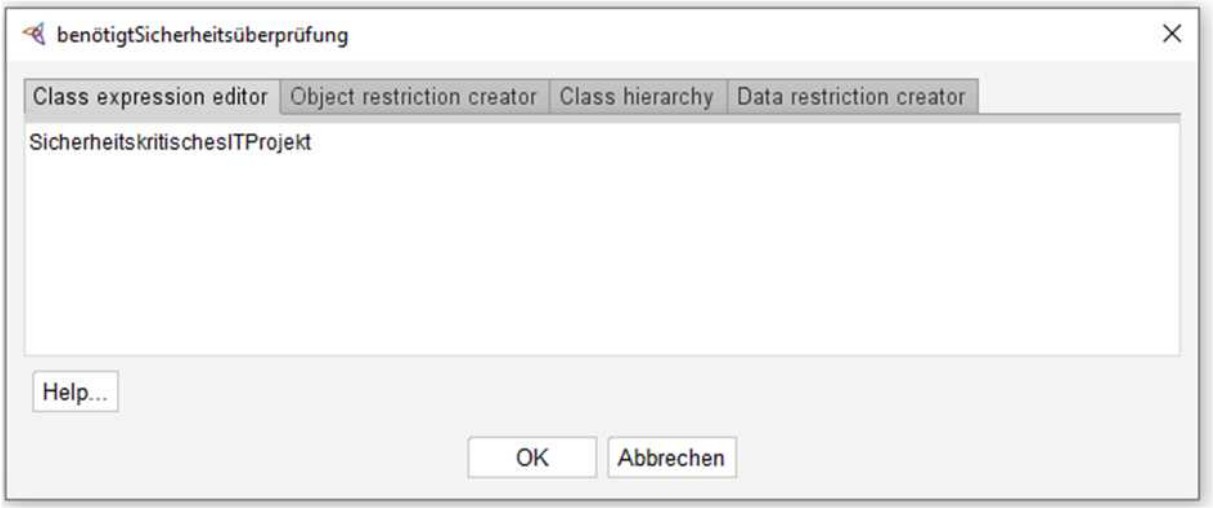


Abbildung 48:Class Expression Editor für die Zuweisung einer Domäne zu einem Attribut

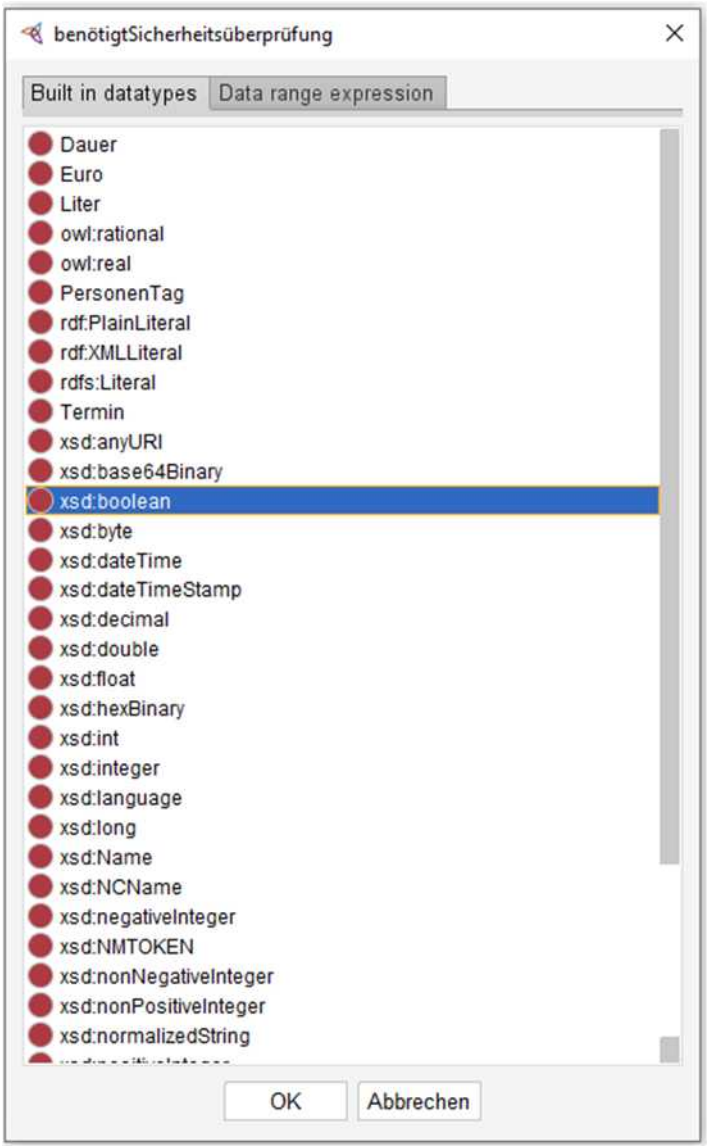


Abbildung 49: Zuweisung eines Datentyps im Nachbereich des Attributs

Protégé unterstützt die nachfolgenden primitiven Datentypen für die Zuordnung:

Datentyp	**Erläuterung**
String	Der Datentyp „String“ wird angewendet, wenn ein Attribut als Wert eine Zeichenkette annimmt.
Float	Der Datentyp „Float“ wird benutzt, wenn ein Attribut als Wert eine Gleitkommazahl annimmt.
Integer	Der Datentyp „Integer“ wird verwendet, wenn ein Attribut als Wert eine ganze Zahl annimmt.
Boolean	Der Datentyp „Boolean“ wird benutzt, wenn ein Attribut als Wert einen Wahrheitswert (wahr, falsch) annimmt.

Tabelle 40: Erläuterung der primitiven Datentypen

In Protégé besteht darüber hinaus die Möglichkeit, eigene Datentypen anzulegen. Die zugrunde gelegte PM-Domänen-Ontologie bietet individuelle Datentypen wie Dauer, Euro und Liter an. In der sicherheitskritischen IT-Projekt-Ontologie werden jedoch keine individuellen Datentypen angelegt. Dies liegt zum einen daran, dass das CBR-Tool jCORA ohne Anpassungen im Quellcode des Programms keine zusätzlich angelegten Datentypen unterstützt. Zum anderen können die meisten Attribute mit den primitiven Datentypen abgebildet werden. Diese Sichtweise wird auch in DEBELLIS (2021), S. 48, verfolgt, wo die Verwendung der primitiven Datentypen den Regelfall darstellt. Sollte es dennoch notwendig sein, einen eigenständigen Datentyp zu erstellen, empfiehlt sich die Konstruktion einer neuen Klasse.

Anhand der Klasse *Vergabeverfahren* werden exemplarische Konstruktionsentscheidungen für die Attribute begründet. Die nachfolgende Abbildung 50 zeigt die Klasse *Vergabeverfahren* mit ihren Attributen und mit den zugewiesenen Datentypen.

Abbildung 50: Die Klasse „*Vergabeverfahren*“ mit den zugewiesenen Attributen

Die Konstruktion der Attribute für die Klasse *Vergabeverfahren* orientiert sich an den öffentlichen Bekanntmachungen von öffentlichen Ausschreibungen und wurde durch zusätzliche Attribute erweitert. Nachfolgend werden die Attribute der Klasse *Vergabeverfahren* hinsichtlich ihrer Konstruktionsentscheidungen erläutert.

- Das Attribut *hatCPVCode* drückt die eindeutige Identifizierungsnummer der Ausschreibung aus. Das Attribut *hatCPVCode* hat den Datentyp String zugewiesen bekommen. Der CPV-Code ist zwar eine Zahlenkombination, jedoch wird die letzte Zahl des Codes durch das Zeichen - getrennt. Außerdem werden bei einer Integer-Zuweisung führende Nullen nicht berücksichtigt. Diese könnten jedoch den CPV-Code verfälschen. Daher wird der Datentyp String für die Zuweisung gewählt.
- Das Attribut *wurdeBereitsAufgehoben* drückt aus, ob das Vergabeverfahren bereits in der Vergangenheit ausgeschrieben und während des Vergabeverfahrens aufgehoben wurde. Der Datentyp des Attributs *wurdeBereitsAufgehoben* ist Boolean, da die Angabe als Wahrheitswert ausreicht.

- Das Attribut *hatLinkZuBekanntmachung* repräsentiert den Verweis zu einer Internetadresse. Mittels des Aufrufs der Internetadresse in einem Webbrowser können die Informationen zur öffentlichen Bekanntmachung des Vergabeverfahrens eingesehen werden. Der Datentyp des Attributs *hatLinkZuBekanntmachung* ist ein String, da der Link eine Zeichenkette darstellt, die auf eine Webseite verweist.
- Das Attribut *hatVorgeschaltetenTeilnahmeantrag* drückt die Eigenschaft aus, dass vor dem Vergabeverfahren ein Teilnahmeantrag vorgeschaltet wurde. Der Datentyp des Attributs *hatVorgeschaltetenTeilnahmeantrag* ist ein Boolean. Der Wahrheitswert als Boolean reicht für die Darstellung dieser Eigenschaft aus, da entweder ein vorangegangener Teilnahmeantrag vorhanden ist oder nicht.
- Das Attribut *hatErfüllungsOrt* gibt den Ort an, wo die Leistung des Vergabeverfahrens erbracht werden soll. Der Datentyp des Attributs ist ein String, da die Ortsbezeichnung eine Zeichenkette darstellt.
- Das Attribut *hatArtDesAuftrags* gibt die Klassifikation der angefragten Leistung des Vergabeverfahrens an. Der Datentyp des Attributs ist ein String, da die Klassifikation der angefragten Leistung eine Zeichenkette darstellt, wie beispielweise „Dienstleistungen“.
- Das Attribut *hatLose* gibt an, ob das Vergabeverfahren in mehrere „Lose“ aufgeteilt wurde. Der Datentyp des Attributs *hatLose* ist Boolean, da ein Wahrheitswert ausreicht, um darzustellen, ob das Vergabeverfahren Lose besitzt.
- Das Attribut *hatGeschätztenWert* gibt den geschätzten Wert des Vergabeverfahrens aus Sicht des Auftraggebers an. Der Datentyp des Attributs ist ein Integer-Wert, da die Angabe des geschätzten Wertes keine Gleitkommazahl, sondern eine geschätzte ganze Zahl ist.
- Das Attribut *hatBezeichnungDesAuftrags* gibt den Namen des Vergabeverfahrens an, welcher mittels einer Zeichenkette angegeben wird und daher einen String als Datentyp zugewiesen erhalten hat.
- Das Attribut *hatLaufzeitDerAnfgefragtenLeistung* gibt die Laufzeit der angefragten Leistung des Vergabeverfahrens an. Der Datentyp des Attributs ist, unter der Voraussetzung der Maßeinheit für die Angabe der Jahre, „Integer“, da die Laufzeit als eine ganze Zahl ausgedrückt wird.
- Das Attribut *hatNUTSCode* drückt den Code „Nomenclature des unités territoriales statistiques“ aus, welcher ein zusammengesetzter Schlüssel aus Ziffern und Buchstaben ist; vgl. EUROSTAT (2022), S. 4. Der Zweck des NUTS-Codes ist es, eine eindeutige Aufteilung der räumlichen Gebiete der amtlichen Statistik in der Europäischen Union vornehmen zu können. Ein NUTS-Code ist bei Ausschreibungen in der Europäischen Union zwingend anzugeben. Der Datentyp des Attributs *hatNUTSCode* ist ein String, da es eine Kombination aus Zahlen und Buchstaben ist. Eine Liste der aktuellen NUTS-

Codes für Deutschland kann in EUROSTAT (2022), S. 30-40, eingesehen werden. Mittels des NUTS-Codes kann ein eindeutiger geographischer Bezug zu einer Ausschreibung in der Europäischen Union hergestellt und nach bestimmten Ausschreibungen in einer bestimmten Region der Europäischen Union gesucht werden. Der NUTS-Code erleichtert Auftragnehmern die Suche nach Aufträgen, da sie mittels der NUTS-Klassifikation das Gebiet vereinfacht eingrenzen können.

- Das Attribut *hatArtDesÖffentlichenAuftraggeber* gibt eine Klassifikation des öffentlichen Auftraggebers an. Die Klassifikation ist als Zeichenkette definiert und wird daher dem Attribut *ArtDesÖffentlichenAuftraggebers* als Datentyp String zugewiesen.
- Das Attribut *hatHaupttätigkeiten* gibt die Haupttätigkeiten des öffentlichen Auftraggebers an, wie beispielsweise „Öffentliche Sicherheit und Ordnung“. Die Angabe der Haupttätigkeit ist eine Zeichenkette. Dem Attribut wird als Datentyp String zugewiesen.
- Das Attribut *überschreitetSchwellenwert* gibt an, ob der Auftragswert den Schwellenwert für die Anwendung des europäischen Vergaberechts erreicht. Der Datentyp Boolean eignet sich, um die Angabe, ob der Schwellenwert erreicht wurde, anzugeben.
- Das Attribut *hatVergabeanwalt* gibt an, ob die ausschreibende Stelle einen Vergabeanwalt für das Vergabeverfahren beauftragt hat. Die Angabe erfolgt mittels des Datentyps Boolean, da die Information ausreicht, ob eine vergaberechtliche anwaltliche Begleitung für das Vergabeverfahren existiert oder nicht.
- Das Attribut *istKomplexesVergabeverfahren* gibt an, ob das Vergabeverfahren als komplexes Vergabeverfahren eingestuft wird. Die Angabe erfolgt mittels des Datentyps Boolean, da mittels eines Wahrheitswerts eine Einstufung für die Komplexität des Vergabeverfahrens vorgenommen werden kann.
- Das Attribut *istBundesausschreibung* gibt an, ob die Ausschreibung eine Leistung ausschreibt, die für die gesamte Bundesrepublik Deutschland relevant und somit unabhängig von einem Bundesland ist. Die Angabe erfolgt mittels des Datentyps Boolean.
- Das Attribut *hatArtDesAuftraggebers* spezifiziert die Art des Auftraggebers, welcher im Attribut *hatArtDesÖffentlichenAuftraggebers* angegeben wird, und ermöglicht somit die Identifizierung der spezifischen Nutzer der angeforderten Leistung durch die Nennung der entsprechenden Behörde oder Organisation mit Sicherheitsaufgabe. Zur Speicherung dieser Information wird der Datentyp String verwendet.
- Das Attribut *hatTCVPlan* spezifiziert den geplanten Auftragswert aus Sicht eines potenziellen Auftragnehmers, der ein Bieter ist. Der geplante Auftragswert aus Sicht des Auftragnehmers kann höher oder niedriger sein als der durch den Auftraggeber geschätzte Auftragswert (durch das Attribut *hatGeschätztenWert* ausgedrückt). Das Attribut *hatTCVPlan* wird dem Datentyp Integer zugeordnet, da es sich um eine ganze Zahl handelt, die den geplanten Auftragswert ausdrückt.

- Das Attribut *betrifftBundesland* gibt an, welches Bundesland von dieser Ausschreibung betroffen ist. Die Angabe erfolgt mittels eines Strings, da die Bezeichnung des Bundeslands eine Zeichenkette darstellt. Die Zuordnung zu einem Bundesland kann mithilfe der Attribute *hatNUTSCode* und *hatErfüllungsOrt* automatisiert erfolgen in der Form, dass eine SWRL-Regel erstellt wird, die dieses Attribut automatisch mit den Werten aus *hatNUTSCode* und *hatErfüllungsOrt* befüllt.

Eine vollständige Darstellung der Attribute der sicherheitskritischen IT-Projekt-Ontologie kann SETHUPATHY (2024), S. 267-291, entnommen werden.

3.2.3.7 Konstruktion der Kardinalitäten

Nachfolgend wird auf die Konstruktion der Kardinalitäten von nicht-taxonomischen Relationen und Attributen eingegangen. Die Kardinalitäten werden in einer tabellarischen Darstellung angegeben, welche wie folgt aufgebaut ist:

Domain	**Name**	**Range**	**Kardinalität**
Vergabeverfahren	hatLose	Boolean	only

Tabelle 41: Tabellenstruktur für die Erläuterung von Kardinalitäten

Die Tabelle 441 zeigt beispielhaft eine Kardinalität, die angibt, dass eine Instanz der Klasse *Vergabeverfahren* genau einen („only“) Attributswert des Attributs *hatLose* hat, welcher dem Datentyp „Boolean“ zugeordnet ist. Ein Gesamtüberblick über alle Kardinalitäten sowohl der nicht-taxonomischer Relationen als auch der Attribute der sicherheitskritischen IT-Projekt-Ontologie findet sich in SETHUPATHY (2024), S. 681-697 bzw. S. 698-722.

Die Konstruktion der Kardinalitäten erfolgt in Protégé im Bereich der Klassen im Feld „SubClass Of“ durch das Klicken des Feldes „Add“, wie die nachfolgende Abbildung51 zeigt.

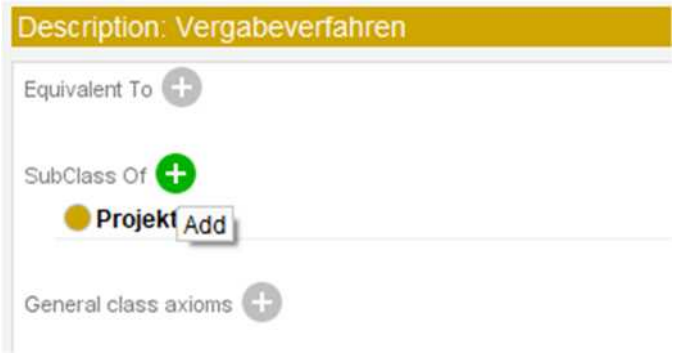


Abbildung 51: Aufrufen der Funktion „SubClass Of“

Im Class Expression Editor wird die Kardinalität eingegeben, indem die Kardinalität einer Eigenschaft (Attribut oder nicht-taxonomische Relation) spezifiziert wird. Das zuvor genannte

Beispiel, dass die Klasse *Vergabeverfahren* für das Attribut *hatLose* genau einen Attributswert mit dem Datentyp Boolean besitzt, wird im Class Expression Editor wie folgt ausgedrückt:

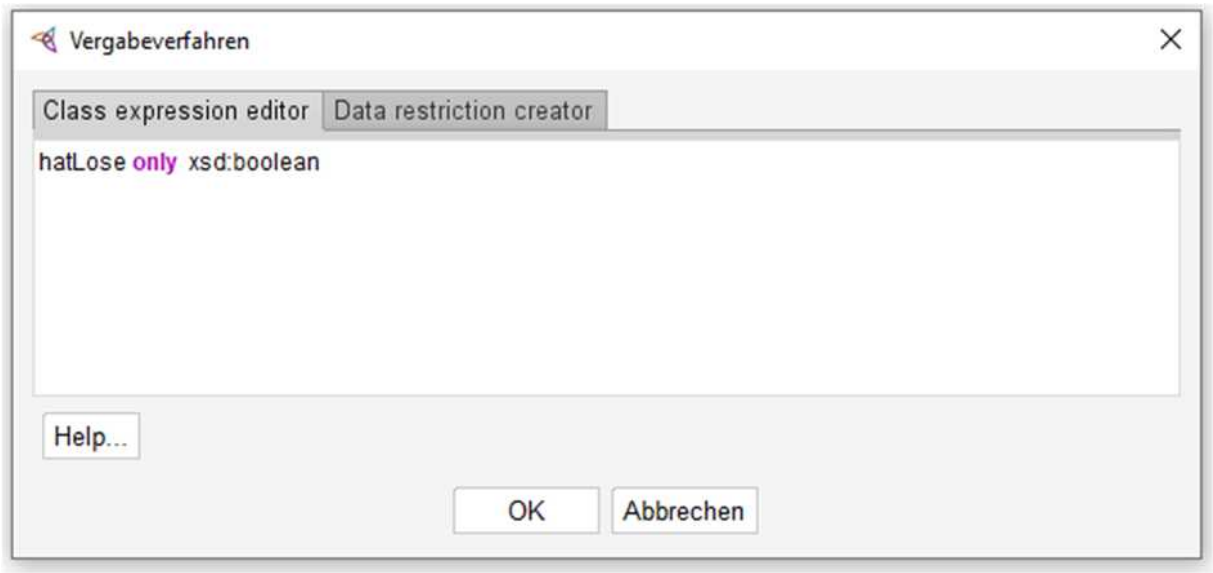


Abbildung 52: Spezifizierung der Kardinalität im Class Expression Editor

Protégé bietet verschiedene Kardinalitäten an, die am Beispiel der nicht-taxonomischen Relation *hatAusschlusskriterium* veranschaulicht werden. Die nicht-taxonomische Relation *hatAusschlusskriterium* ist in der Domäne der Klasse *Vergabeverfahren* und in der Range der Klasse *Ausschlusskriterium* zugeordnet.

Kardinalität	**Erläuterung**
some	Jede Instanz, die der Klasse *Vergabeverfahren* zugeordnet ist, hat für die Relation *hatAusschlusskriterium* mindestens eine Instanz der Klasse *Ausschlusskriterium*.
only	Jede Instanz, die der Klasse *Vergabeverfahren* zugeordnet ist, hat für die Relation *hatAusschlusskriterium* genau eine Instanz der Klasse *Ausschlusskriterium*.
min X	Jede Instanz, die der Klasse *Vergabeverfahren* zugeordnet ist, hat für die Relation *hatAusschlusskriterium* mindestens X Instanzen der Klasse *Ausschlusskriterium*. Die Kardinalität „min 1“ ist äquivalent zu der Kardinalität „some“.
exactly X	Jede Instanz, die der Klasse *Vergabeverfahren* zugeordnet ist, hat für die Relation *hatAusschlusskriterium* genau X Instanzen der Klasse *Ausschlusskriterium*. Die Kardinalität „exactly 1“ ist äquivalent zur Kardinalität „only“.
max X	Jede Instanz, die der Klasse *Vergabeverfahren* zugeordnet ist, hat für die Relation *hatAusschlusskriterium* maximal X Instanzen der Klasse *Ausschlusskriterium*.

Abbildung 53: Erläuterung von Kardinalitäten

Anhand der Klasse *Vergabeverfahren* werden die Kardinalitäten für die Eigenschaften begründet. Die nachfolgende Abbildung54 zeigt die Klasse *Vergabeverfahren* mit ihren Eigenschaften

– Attributen und nicht-taxonomischen Relationen – sowie mit den zugewiesenen Kardinalitäten.

Abbildung 54: Kardinalitäten der Attribute und der nicht-taxonomischen Relationen der Klasse „*Vergabeverfahren*“

Die Klasse *Vergabeverfahren* umfasst folgende Attribute: *hatCPVCode*, *wurdeBereitsAufgehoben*, *hatLinkZuBekanntmachung*, *hatVorgeschaltetenTeilnahmeantrag*, *hatErfüllungsOrt*, *hatArtDesAuftrags*, *hatLose*, *hatGeschätztenWert*, *hatBezeichnungDesAuftrags*, *hatLaufzeitDerAngefragtenLeistung*, *hatNUTSCode*, *hatArtDesÖffentlichenAuftraggebers*, *hatHaupttätigkeiten*, *überschreitetSchwellenwert*, *hatVergabeanwalt*, *istKomplexesVergabeverfahren* sowie *istBundesausschreibung*, *hatArtDesAuftragsgebers*, *hatTCVPlan*, *betrifftBundesland*.

Die Klasse *Vergabeverfahren* umfasst folgende nicht-taxonomische Relationen: *hatVergabeverfahrenArt*, *hatAusschlusskriterium*, *hatZuschlagskriterium*, *hatEignungskriterium* sowie *hatVergabeverfahrenEntscheidung*

Nachfolgend wird zunächst auf die Kardinalitäten der Attribute eingegangen und anschließend auf die Kardinalitäten der nicht-taxonomischen Relationen der Klasse *Vergabeverfahren*.

- Das Attribut *hatCPVCode* wird mit der Kardinalität „only String“ angegeben, da es sich um eine eindeutige Identifizierungsnummer handelt, die nur einmalig und eindeutig für ein Vergabeverfahren vergeben wird.
- Das Attribut *hatNUTSCode* wird mit einer Kardinalität von „some String“ versehen, da für jede Ausschreibung mehrere geographische Gebiete mittels des NUTS-Codes spezifiziert werden können. Gemäß den Vorschriften für europäische Ausschreibungen ist die Angabe eines CPV-Codes und eines NUTS-Codes verpflichtend. Um sicherzustellen, dass diese Anforderung erfüllt wird, wird im späteren Verlauf eine SWRL-Regel konstruiert. Diese Regel legt fest, dass die Angabe eines NUTS-Codes, ausgedrückt durch das Attribut *hatNUTSCode*, erforderlich ist, wenn ein CPV-Code, ausgedrückt durch das Attribut *hatCPVCode*, angegeben wurde.
- Das Attribut *wurdeBereitsAufgehoben* ist mit der Kardinalität „only Boolean“ versehen, um anzugeben, dass es nur einen Boolean-Wert geben kann. Diese Entscheidung wird damit begründet, dass die Angabe, ob das Vergabeverfahren bereits in der Vergangenheit aufgehoben und neu ausgeschrieben wurde, durch einen einzigen Boolean-Wert ausreichend repräsentiert.
- Das Attribut *hatLinkZuBekanntmachung* wird mit der Kardinalität „min 1 String“ angegeben. Dies liegt daran, dass es mindestens einen Link zu der Bekanntmachung des Vergabeverfahrens geben muss. Es kann jedoch vorkommen, dass die Bekanntmachung auf mehreren Vergabeseiten veröffentlicht wurde. Aufgrund dieser Möglichkeit der Veröffentlichung auf verschiedenen Webseiten wurde die Kardinalität für das Attribut *hatLinkZurBekanntmachung* auf „min 1 String“ festgelegt.
- Das Attribut *hatVorgeschaltetenTeilnahmeantrag* wird mit der Kardinalität „only Boolean“ angegeben. Ein Vergabeverfahren kann nur genau einen vorgeschalteten Teilnahmeantrag besitzen. Mehrere Teilnahmeanträge sind für ein konkretes Vergabeverfahren nicht vorgesehen. Daher ist die Kardinalität „only Boolean“ für das Attribut gewählt worden.
- Das Attribut *hatErfüllungsOrt* wird mit der Kardinalität „min 1 String“ angegeben, da es mindestens einen Erfüllungsort für ein Vergabeverfahren geben muss, aber auch an mehreren Orten die Leistung erbracht werden kann. Dies kann beispielsweise der Fall sein, wenn ein Vergabeverfahren in verschiedenen Losen ausgeschrieben wird, in denen die Leistungen an unterschiedlichen Orten erbracht werden müssen.
- Das Attribut *hatArtDesAuftrags* wird mit der Kardinalität „only String“ angegeben, da es nur genau eine Bezeichnung für die Art des Auftrags geben darf.

- Das Attribut *hatLose* wird mit der Kardinalität „only Boolean“ angegeben, da eine einmalige Angabe eines Boolean-Werts ausreicht, um auszudrücken, ob das Vergabeverfahren in mehrere Losen unterteilt wurde.
- Das Attribut *hatGeschätztenWert* wird mit der Kardinalität „min 1 Integer“ angegeben. Diese Konstruktion begründet sich damit, dass es für jedes Vergabeverfahren mindestens einen geschätzten Wert durch die Vergabestelle geben muss, um entscheiden zu können, ob unter- oder oberschwellig ausgeschrieben werden muss. Jedoch können darüber hinaus weitere geschätzte Werte vorliegen, wenn das Vergabeverfahren mehrere Lose besitzt.
- Das Attribut *hatBezeichnungDesAuftrags* wird mit der Kardinalität „only String“ angegeben. Diese Konstruktion begründet sich dahingehend, dass für ein Vergabeverfahren eine eindeutige Bezeichnung für den Auftragsgegenstand definiert werden muss. Mehrere Bezeichnungen für einen Auftrag sind für Vergabeverfahren nicht vorgesehen.
- Das Attribut *hatLaufzeitDerAnfgefragtenLeistung* wird mit der Kardinalität „min 1 Integer“ angeben. Es wird mindestens eine fest definierte Laufzeit für die ausgeschriebene Leistung angegeben. Wenn die Leistung in mehreren Losen ausgeschrieben wurde, können auch mehrere Laufzeiten für die Lose angegeben werden.
- Das Attribut *hatArtDesÖffentlichenAuftraggeber* wird mit der Kardinalität „only String“ angegeben, da genau eine Behörde genannt wird, welche die Leistung ausschreibt.
- Das Attribut *hatHaupttätigkeit* wird mit der Kardinalität „only String“ konstruiert, da in den Bekanntmachungen nur genau eine Haupttätigkeit genannt wird.
- Das Attribut *ueberschreitetSchwellenwert* wird mit der Kardinalität „only Boolean“ konstruiert. Die Angabe, ob das Vergabeverfahren den Schwellenwert überschreitet, ist nur einmalig in einem Vergabeverfahren anzugeben.
- Das Attribut *hatVergabeanwalt* wird mit der Kardinalität „only Boolean“ konstruiert. Das Attribut drückt aus, ob für das Vergabeverfahren ein Vergabeanwalt hinzugezogen wurde. Für diesen genannten Fall reicht ein Boolean-Wert aus, um dieses sprachliche Ausdrucksmittel für sicherheitskritische IT-Projekte bereitzustellen.
- Das Attribut *istKomplexesVergabeverfahren* wird mit der Kardinalität „only Boolean“ konstruiert. Das Attribut ist eine einmalige Angabe, ob das Vergabeverfahren als komplexes Vergabeverfahren eingestuft wird, und wird für die spätere automatisierte Einstufung mittels SWRL-Regeln benötigt.

- Das Attribut *istBundesausschreibung* wird mit der Kardinalität „only Boolean“ konstruiert, da eine einmalige Angabe ausreicht, um anzugeben, ob das Vergabeverfahren auf Bundesebene oder auf ein bestimmtes Bundesland bezogen ist. Es wird später verwendet, um mittels SWRL-Regeln eine Ableitung für diese Attribut aus den NUTS-Codes (Attribut: *hatNUTSCode*) zu treffen.
- Das Attribut *betrifftBundesland* wird mit der Kardinalität „only String“ konstruiert, da für ein Vergabeverfahren genau ein Bundesland als Betroffener der ausgeschriebenen Leistung definiert wird. Bei einer Bundesausschreibung wird der Wert „Deutschland“ zugewiesen, um darzustellen, dass alle Bundesländer betroffen sind. Auch diese Zuweisung wird im späteren Verlauf durch eine SWRL-Regel automatisiert vorgenommen.
- Die nicht-taxonomische Relation *hatAusschlusskriterium* wird mit der Kardinalität „min 1 *Ausschlusskriterium*“ konstruiert.
- Die nicht-taxonomische Relation *hatZuschlagskriterium* wird mit der Kardinalität „min 1 *Zuschlagskriterium*“ konstruiert. Als Zuschlagskriterium bei einem Vergabeverfahren kann beispielsweise der Preis betrachtet werden, wenn er als ausschließliches Zuschlagskriterium gewählt wird. Es muss mindestens ein Kriterium vorliegen, um einen Zuschlag vornehmen zu können.
- Die nicht-taxonomische Relation *hatEignungskriterien* wird mit der Kardinalität „min 1 *Eignungskriterium*“ konstruiert.
- Die nicht-taxonomische Relation *hatVergabeverfahrenArt* wird mit der Kardinalität „only *VergabeverfahrenArt*“ konstruiert. Diese Konstruktion wird damit begründet, dass für ein Vergabeverfahren genau eine Vergabeart vorgesehen ist.
- Die nicht-taxonomische Relation *hatVergabeverfahrenEntscheidung* wird mit der Kardinalität „min 2 *VergabeverfahrenEntscheidung*“ konstruiert. Diese Konstruktion ermöglicht es, dass bei einer freihändigen Vergabe mindestens zwei Entscheidungen getroffen werden müssen, nämlich welche Art von Vergabeverfahren angewendet werden soll und wer den Zuschlag erhalten wird.

3.2.3.8 Konstruktion von Semantic Web Rules

Nachfolgend wird auf die Konstruktion von Semantic Web Rules eingegangen. Die Semantic Web Rules werden in einer tabellarischen Darstellung angegeben, die wie folgt aufgebaut ist:

Regel	**natürlichsprachliche Übersetzung**

Tabelle 42: Tabellenstruktur für die Erläuterung von Semantic Web Rules

In der Spalte „Regel" wird eine Semantic Web Rule (SWR) in der Ausdrucksweise aus Protégé dargestellt. In der Spalte „natürlichsprachliche Übersetzung" wird der Regelinhalt natürlichsprachlich erläutert.

Die Semantic Web Rule Language (SWRL) ist die Sprache, in der die nachfolgenden Regeln definiert werden. Sie ist eine Kombination aus OWL DL und Unary / Binary Datalog RuleML. Eine Regel besteht aus einer Antezedensbedingung (antecedent) und einer Konsequenz (consequent): „antecedent → consequent".

Zwei Ausdrücke (Atome), die in einer SWRL-Regel aufeinanderfolgen, werden mittels des Zeichens „^" getrennt. Der Antezedens- und der Konsequenz-Teil einer Regel können aus 0 bis n Ausdrücken bestehen, wobei jeder Ausdruck in folgender Form dargestellt werden kann: $C(x)$, $P(x,y)$, $contains(x,y)$, $greaterthan\ (x,y)$. Diese Formen sind wie folgt zu verstehen: C ist eine Klasse, P ist eine Eigenschaft, x,y sind Variablen oder können Instanzen darstellen, $contains()$ und $greaterthan()$ sind Hilfsfunktionen, die in SWRL vordefiniert sind.

Eine Regel wird genau dann ausgelöst, wenn jeder Ausdruck in der Antezedensbedingung erfüllt ist. Wenn die Antezedensbedingung erfüllt sind, folgt die Konsequenz. Im Folgenden wird eine Regel anhand eines Beispiels erläutert, das aus folgenden Ausdrücken besteht:

- SicherheitskritischesITSystem(?its)
- hatSchutzbedarfskategorie (?its, ?s)
- swrlb:contains(„sehr hoch")
- MitProjektbezugMitarbeiter(?p)
- hatErweiterteSicherheitsüberpruefung(?p, true)

Ausdrücke, die mit einem „?" beginnen, wie beispielsweise „?its", beziehen sich auf die Instanzen der Klassen. In diesem Beispiel bezieht sich die Variable „?its" auf alle Instanzen der Klasse *SicherheitskritischesITSystem*.

Die Regel ist wie folgt konstruiert:

Antezedensbedingung		**Konsequenz**
SicherheitskritischesITSystem(?its) ^ hatSchutzbedarfskategorie (?its, ?s) ^ swrlb:contains(?s, „sehr hoch") ^ MitProjektbezugMitarbeiter(?p)	→	hatErweiterteSicherheitsueberpruefung(?p, true)

Tabelle 43: Darstellung einer exemplarischen SWRL-Regel

Diese Regel drückt exemplarisch aus, dass die Projektmitarbeiter eine erweiterte Sicherheitsüberprüfung besitzen müssen, wenn ein sicherheitskritisches IT-System eine sehr hohe Schutzbedarfskategorie hat: Wenn ein sicherheitskritisches IT-System die Schutzbedarfskategorie „sehr hoch“ hat und Instanzen zu der Klasse *MitProjektbezugMitarbeiter* existieren, dann wird die Eigenschaft *hatErweiterteSicherheitsueberpruefung* der Klasse *MitProjektbezugMitarbeiter* auf „true“ gesetzt.

Für die Verwendung von Semantic Web Rules ist in Protégé die Verwendung des Pellet Reasoners notwendig. Bei der Verwendung eines anderes Reasoners erfolgt eine Fehlermeldung. Die Verwendung des Pellet Reasoners für die Semantic Web Rules wird beispielsweise von DEBELLIS (2021), S. 15, SYCHEV/ANIKIN/DENISOV (2021), S. 472, und BATSAKIS/TACHMAZIDIS/ANTONIOU (2017), S. 25, vorgeschlagen. Neuere Entwicklungen zeigen aber auch, dass Apache Jena als Reasoner für SWRL als vorteilhafter hinsichtlich der Laufzeit und bei der Nutzung von verschiedenen Programmiersprachen angesehen werden kann; vgl. SYCHEV/ANIKIN/DENISOV (2021), S. 480.

Die Konstruktion von Semantic Web Rules Language erfolgt mittels eines Plug-Ins in Protégé. Der Aufruf der Konstruktionshilfe „SWRL-Tab“ erfolgt durch Window>Tabs>SWRLTab. Durch die Auswahl wird folgendes Fenster in der Abbildung 55 geöffnet, um eine Semantic Web Rule anzulegen.

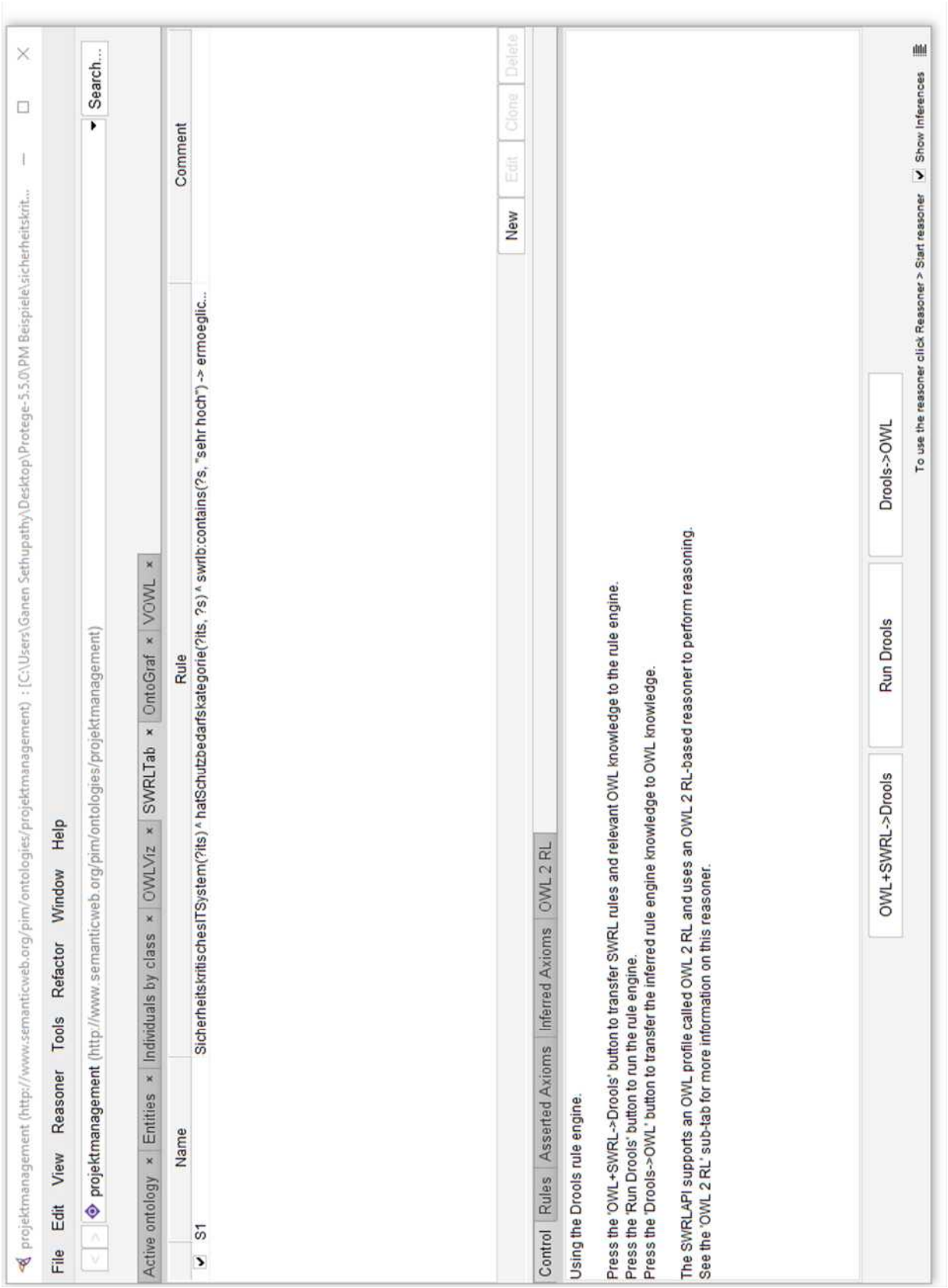


Abbildung 55: SWRL-Tab in Protégé

Für die Konstruktion einer Semantic Web Rule sind verschiedene Ausdrücke für die Antezedensbedingung und für die Konsequenz zu berücksichtigen, wie anhand der nachfolgenden Regel auf exemplarische Art gezeigt wird.

Regel	**natürlichsprachliche Übersetzung**
SicherheitskritischesITSystem(?its) ^ hatSchutzbedarfskategorie(?its, ?s) ^ swrlb:contains(?s, „sehr hoch“) → ermoeglichtNearshore(?its, false) ^ ermoeglichtOffshore(?its, false)	Wenn ein sicherheitskritisches IT-System die Einstufung der Schutzbedarfskategorie „sehr hoch“ hat, dann ist sowohl kein Nearshore als auch kein Offshore möglich.

Tabelle 44: SWRL-Regel mit ihrer natürlichsprachlichen Übersetzung

Zunächst wird die Klasse *SicherheitskritischesITSystem* mit der Variable „?its” deklariert, sodass alle Instanzen der Klasse *SicherheitskritischesITProjekt* in dem Ausdruck „SicherheitskritischesITProjekt(?its)“ berücksichtigt werden sollen. Dieser Ausdruck wird mittels „^“ mit dem Ausdruck „hatSicherheitsstufe(?its, ?s)” gebunden. Durch den Ausdruck „hatSicherheitsstufe(?its, ?s)“ wird auf das Attribut *hatSicherheitsstufe* der Klasse *SicherheitskritischesITProjekt* verwiesen. Dabei steht die Variable „?s“ für das Attribut *hatSicherheitsstufe* und wird mittels der vordefinierten Hilfsfunktion *swlb:contains* auf den Wert „sehr hoch“ überprüft. Falls die Antezedensbedingung der Regel erfüllt ist, werden die Attribute *ermoeglichtNearshore* und *ermöglichtOffshore* in der Konsequenz der Regel auf „false“ gesetzt.

Die erläuterte Semantic Web Rule repräsentiert die Regel, dass für ein sicherheitskritisches IT-System keine Entwicklungsleistung außerhalb von Deutschland erfolgen darf, wenn seine Schutzbedarfskategorie als „sehr hoch“ eingestuft wurde.

Die zuvor erläuterte Konstruktion einer Semantic Web Rule erfolgt im SWRL-Tab von Protégé, wie in der nachfolgenden Abbildung56 dargestellt wird.

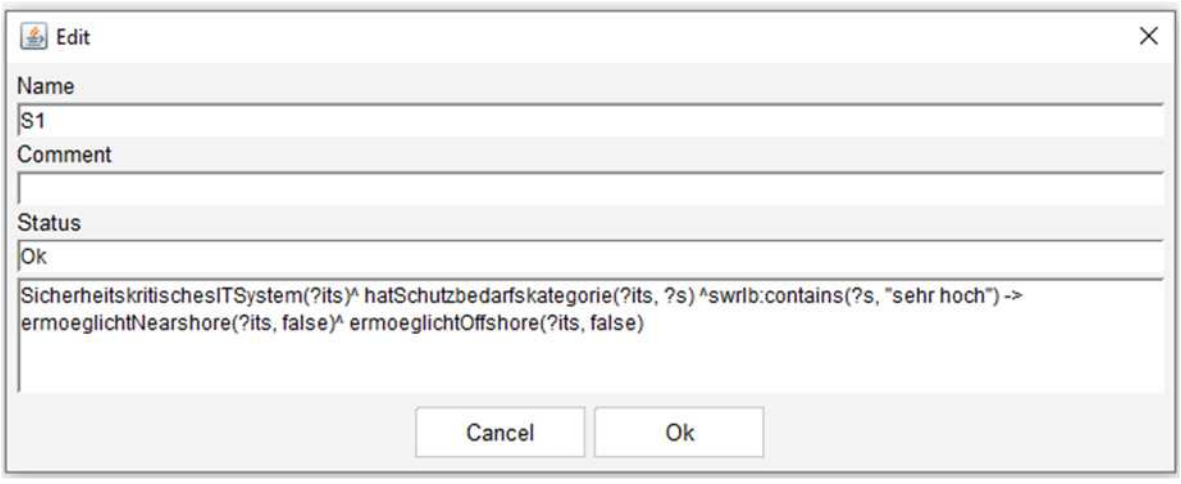


Abbildung 56: Konstruktion einer Semantic Web Rule in Protégé

Bei einer fehlerhaften Konstruktion bleibt der „Ok“-Button ausgegraut, sodass eine Bestätigung nicht erfolgen kann. Des Weiteren wird eine Fehlermeldung im Feld „Status” ausgegeben, damit eine Fehleranalyse stattfinden kann. Bei einer fehlerfreien Konstruktion wird im Feld Status „Ok“ angezeigt, wie die Abbildung56 zeigt.

Im Folgenden werden die Semantic Web Rules für die Klasse *Vergabeverfahren* hinsichtlich ihrer Konstruktion erläutert.

Die in Tabelle45 dargestellte Regel besagt, dass der Schwellenwert erreicht wird, wenn der geschätzte Wert der Vergabe größer oder gleich 250.000 Euro ist. Durch diese Modellierung ist es möglich, anhand des geschätzten Werts eine Regel festzulegen, ob die Vergabe im unterschwelligen oder oberschwelligen Bereich erfolgt. Bei der Konstruktion dieser Regel wird die Hilfsfunktion *swrlb:greaterThanOrEqual* verwendet, welche die Fallprüfung für den Integer-Werts 250.000 durchführt.

Regel	**natürlichsprachliche Übersetzung**
Vergabeverfahren(?v) ^ hatGeschaetztenWert(?v, ?np) ^ swrlb: greaterThanOrEqual(?np, 250000) → ueberschreitetSchwellenwert(?v, true)	Wenn der geschätzte Wert eines Vergabeverfahrens 250.000 Euro erreicht, wird der Schwellenwert erreicht.

Tabelle 45: Regel zur Überprüfung des Schwellenwerts

Die nachfolgende Tabelle46 stellt die Regel dar, welche anhand des NUTS-Codes das betroffene Bundesland der Ausschreibung ermittelt. Die Konstruktion dieser Regel ermöglicht die automatisierte Zuweisung des Bundeslandes sowie die Angabe, dass die Ausschreibung keine Bundesausschreibung ist. Diese Konstruktionsentscheidung ist damit begründet, dass die Eindeutigkeit des NUTS-Codes eine automatisierte Zuweisung des Bundeslandes ermöglicht. Dies gilt ebenso bei mehreren NUTS-Codes, wenn mehrere Bundesländer betroffen sind.

Regel	**natürlichsprachliche Übersetzung**
Vergabeverfahren(?v) ^ hatNutsCode(?v, ?nc) ^ swrlb:contains(„DE1“) → betrifftBundesland(?v, „Baden-Württemberg“) ^ istBundesausschreibung(?v, „false“)	Wenn ein Vergabeverfahren den NUTS-Code „DE1“ als Substring beinhaltet, dann wird das Attribut *betrifftBundesland* auf den Wert „Baden-Württemberg“ gesetzt und das Attribut *istBundesausschreibung* auf „false“ gesetzt.

Tabelle 46: Regel zur automatisierten Ableitung des Bundeslandes

Die nachfolgende Tabelle 47 veranschaulicht die Regel, nach der durch das Attribut *hatArtDesAuftraggebers* eines Vergabeverfahrens automatisch die Bezeichnung der erforderlichen Dienstvorschrift für das sicherheitskritische IT-Projekt abgeleitet werden kann. Dies gilt beispielsweise, wenn im Attribut *hatArtDesAuftraggebers* ein Substring wie „Feuerwehr“ als Wert vorhanden ist.

Regel	**natürlichsprachliche Übersetzung**
Vergabeverfahren(?v) ^ SicherheitskritischesITProjekt(?its) ^ erfordertVergabeverfahren(?its, ?v) ^ hatArtDesAuftraggebers(?v, ?ha) ^ swrlb:contains(?ha, „Feuerwehr“) ^ erfordertDienstvorschrift(?its, ?a) → hatBezeichnung(?a, „Feuerwehrdienstvorschrift“)	Wenn es sich bei einem Vergabeverfahren um ein sicherheitskritisches IT-Projekt handelt, bei dem die Feuerwehr als Art des Auftraggebers (*hatArtDesAuftraggebers*) angegeben ist und eine spezifische Dienstvorschrift (*erfordertDienstvorschrift*) verlangt wird, ist es erforderlich, dass diese Dienstvorschrift die Bezeichnung „Feuerwehrdienstvorschrift“ (*hatBezeichnung*) trägt.
Vergabeverfahren(?v) ^ SicherheitskritischesITProjekt(?its) ^ erfordertVergabeverfahren(?its, ?v) ^ hatArtDesAuftraggebers(?v, ?ha) ^ swrlb:contains(?ha, „Polizei“) ^ erfordertDienstvorschrift(?its, ?a) → hatBezeichnung(?a, „Polizeidienstvorschrift“)	Wenn es sich bei einem Vergabeverfahren um ein sicherheitskritisches IT-Projekt handelt, bei dem die Polizei als Art des Auftraggebers (*hatArtDesAuftraggebers*) angegeben ist und eine spezifische Dienstvorschrift (*erfordertDienstvorschrift*) verlangt wird, ist es erforderlich, dass diese Dienstvorschrift die Bezeichnung „Polizeidienstvorschrift“ (*hatBezeichnung*) trägt.

Tabelle 47: Regeln zur automatisierten Ableitung einer Dienstvorschrift

Die nachfolgende Tabelle48 zeigt eine Regel, welche anhand des NUTS-Codes die Relevanz des Vergabeverfahrens als deutschlandweit einstuft. Da der NUTS-Code für Deutschland immer mit DE und mit zwei weiteren Ziffern für die Bundesländer konstruiert wird, ist es an dieser Stelle notwendig zu identifizieren, ob ausschließlich der Begriff „DE“ genannt wird und keine weiterfolgende Ziffernkombination. Dies wird mittels der Hilfsfunktion „swrlb:endsWith(?nc, „DE“) ∧ swrlb:startsWith(?nc, „DE“)“ überprüft. Mittels der Hilfsfunktion „swrlb:startsWith (?nc, „DE“)“ wird überprüft, ob der String mit „DE“ beginnt, und mit der Hilfsfunktion „swrlb:endsWith(?nc, „DE“)“, ob der String mit „DE“ endet. Beide Hilfsfunktionen werden mit einem logischen „und“ verbunden. Sollte dies der Fall sein, wird das Attribut *betrifftBundesland* auf den Wert „Deutschland“ gesetzt und das Attribut *istBundesausschreibung* auf „true“ gesetzt, um zu verdeutlichen, dass es sich bei dem Vergabeverfahren um eine Bundesausschreibung handelt.

Regel	**natürlichsprachliche Übersetzung**
Vergabeverfahren(?v) ^ hatNutsCode(?v, ?nc) ^ swrlb:contains(?nc, „DE“) ^ swrlb:endsWith(?nc, „DE“) ^ swrlb:startsWith(?nc, „DE“) → betrifftBundesland(?v, „Deutschland“) ^ istBundesausschreibung(?v, „true“)	Wenn ein Vergabeverfahren den NUTS-Code „DE“ als Substring beinhaltet und ausschließlich den Begriff DE, dann wird die Eigenschaft „betrifftBundesland“ auf „Deutschland“ gesetzt und die Eigenschaft „istBundesausschreibung“ auf „true“ gesetzt.

Tabelle48: Regel zur automatisierten Ableitung einer Bundesausschreibung

Die nachfolgende Tabelle 49 beinhaltet eine Regel, welche anhand der Eigenschaften (Attribute und nicht-taxonomischen Relationen) eines Vergabeverfahrens:

- ueberschreitetSchwellenwert,
- hatExterneVergabeanwalt,
- wurdeBereitsAufgehoben,
- hatGeschaetztenWert
- hatVergabekostenPlan

ableitet, ob das Vergabeverfahren ein komplexes Vergabeverfahren ist.

Die Auswahl der Eigenschaften für die Auslegung, ob ein Vergabeverfahren komplex ist oder nicht, erweist sich als diskussionswürdig. In diesem Beitrag wurden die genannten Eigenschaften des Vergabeverfahrens aufgrund folgender Begründung ausgewählt:

- Bei Überschreitung des Schwellenwerts müssen umfassende Grundlagen des Vergaberechts berücksichtigt werden.
- Sollte sich die Vergabestelle entscheiden, einen externen Vergabeanwalt für das Vergabeverfahren einzubinden, so kann davon ausgegangen werden, dass komplexere vergaberechtliche Fragestellungen zu klären sind.
- Sollte ein Vergabeverfahren bereits in der Vergangenheit aufgrund unterschiedlicher Beweggründe aufgehoben worden sein, könnte dies ein Hinweis dafür sein, dass das Vergabeverfahren komplex ist (beispielsweise kann ein Vergabeverfahren durch eine Rüge eines unterlegenen Bieters aufgehoben worden sein).
- Bei einer Vergabesumme von über 500.000 Euro ist das Vergabeverfahren höher als der durchschnittliche Vergabewert eines Liefer- oder Dienstleistungsauftrags (Stand 2019) sowie doppelt so hoch wie der Schwellenwert. In dieser Betrachtung werden Bauaufträge nicht berücksichtigt, da diese für sicherheitskritische IT-Projekte unerheblich sind.

- Bei geplanten Vergabekosten (Kosten, die dem Auftraggeber durch das Vergabeverfahren entstehen) von über 50.000 Euro kann davon ausgegangen werden, dass mehrere Abstimmungen mit den Bietern notwendig sind, um das Vergabeverfahren erfolgreich abschließen zu können. Bei den Vergabekosten sind neben den externen Kosten, z. B. für die vergaberechtliche Begleitung, auch die internen Kosten, z. B. für die Abstimmung mit den Bietern, zu berücksichtigen.

Ziel der nachfolgenden Regel ist es, mittels der vorgenannten Eigenschaften eine Einstufung vornehmen zu können, ob das Vergabeverfahren komplex ist.

Regel	**natürlichsprachliche Übersetzung**
Vergabeverfahren(?v) ^ ueberschreitetSchwellenwert(?v, „true“) ^ hatExterneVergabeanwalt(?v, „true“) ^ wurdeBereitsAufgehoben(?v, „true“) ^ hatGeschaetztenWert(?v,?gw) ^ swrlb:greaterThan(?gw, „500000“) ^ hatVergabekostenPlan(?v,?vkp) ^ swrlb:greaterThan(?vkp, „50000“) → istKomplexesVergabeverfahren (?v, „true“)	Wenn ein Vergabeverfahren den Schwellenwert überschreitet, einen externen Vergabeanwalt besitzt, das Vergabeverfahren bereits in der Vergangenheit aufgehoben wurde, einen geschätzte Vergabesumme von über 500.000 Euro besitzt und die Vergabekosten geplant bei über 50.000 Euro liegen, dann ist das Vergabeverfahren als komplex einzustufen.

Tabelle 49: Regel zur automatisierten Ermittlung einer komplexen Ausschreibung

Die nachfolgende Regel in Tabelle50 besagt, dass ein Vergabeverfahren mindestens ein Zuschlagskriterium aufweist, wenn es mindestens ein Ausschlusskriterium umfasst. In anderen Worten: Ein Vergabeverfahren muss bestimmte Anforderungen erfüllen, um in Betracht gezogen zu werden.

Regel	**natürlichsprachliche Übersetzung**
Vergabeverfahren(?v) ^ hatAusschlusskriterium(?v, ?a) → hatZuschlagskriterium(?v, ?b)	Wenn ein Vergabeverfahren mindestens ein Ausschlusskriterium umfasst, dann muss es auch mindestens ein Zuschlagskriterium besitzen.

Tabelle 50: Regel zur automatisierten Ermittlung von Ausschluss- und Zuschlagskriterien

3.2.3.9 Konstruktion von globalen Instanzen

Nachfolgend wird auf die Konstruktion von globalen Instanzen eingegangen. Streng genommen kann die Ansicht vertreten werden, dass Instanzen nicht zu den Komponenten einer Ontologie gehören. Dennoch werden im Folgenden globale Instanzen erläutert und als Komponenten einer Ontologie angesehen, da globale Instanzen in der sicherheitskritischen IT-Projekt-Ontologie genutzt werden, um allgemeingültige Aussagen auszudrücken. Globale Instanzen sollen in konkreten ontologiegestützten CBR-Systemen nicht verändert, auch nicht durch Attribute oder Relationen ergänzt werden. Dieser Auffassung wird im vorliegenden Beitrag gefolgt, um globale Instanzen zu konstruieren und als Ontologiekomponente einzuführen.

Globale Instanzen zeichnen sich im Vergleich zu lokalen Instanzen wie folgt aus:

- Globale Instanzen sind Instanzen, die in der sicherheitskritischen IT-Projekt-Ontologie definiert sind und sich nicht auf ein konkretes Projekt beziehen, sondern projektübergreifend zutreffen.
- Globale Instanzen können als feste Kategorien (z. B. niedrig, mittel, hoch, sehr hoch) definiert werden.
- Globale Instanzen stellen sprachliche Ausdrucksmittel bereit, um zeitliche Konstanten auszudrücken, welche sich für mehrere Projekte gleichzeitig verwenden lassen, wie z. B. konkrete Orte und Farben.
- Konstanten, die immer wieder genutzt werden, können mittels globaler Instanzen spezifiziert und dadurch in einem ontologiegestützten CBR-System automatisch erfasst werden.

Globalen Instanzen werden in folgender tabellarischer Darstellung spezifiziert:

Instanzname:	
Klasse:	
Eigenschaft	**Wert**

- Tabelle 51: Tabellenstruktur für die Spezifizierung der globalen Instanzen

Die tabellarische Darstellung umfasst für die jeweilige globale Instanz den Instanznamen, die Klasse, zu der die Instanz gehört, die instanzbeschreibenden Eigenschaften (Attribute und nicht-taxonomische Relationen) und die zugehörigen Werte der Instanz.

In Protégé werden ausschließlich globale Instanzen angelegt. Protégé bietet verschiedene Möglichkeiten zur Erstellung von (globalen) Instanzen. Die gängigsten Methoden sind:

- im Bereich „Individuals by class“,
- im Bereich „Entities“ unter dem Tab „Classes“ sowie
- im Bereich „Entities“ unter dem Tab „Individuals“.

Im Folgenden wird beispielhaft auf die zuletzt genannte Methode im Bereich „Entities“ unter dem Tab „Individuals“ eingegangen. Die nachfolgende Abbildung57 zeigt den Tab „Individuals“ im Bereich „Entities“.

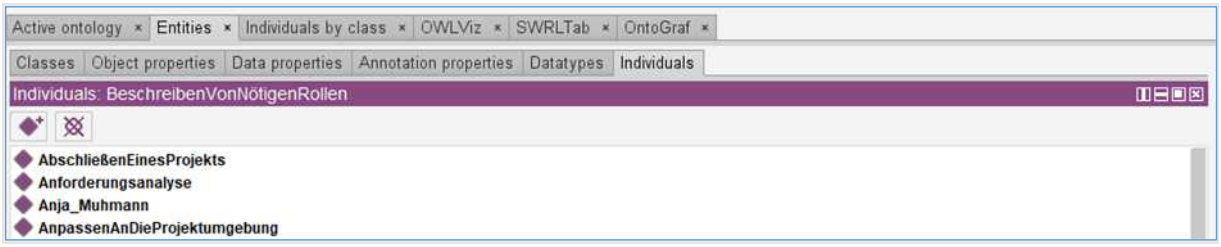


Abbildung 57: Übersicht über die verfügbaren Instanzen

Durch die Auswahl „Add Individual“ wird eine neue Instanz angelegt.

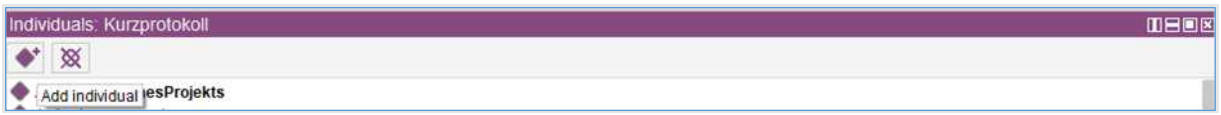


Abbildung 58: Hinzufügen einer neuen Instanz

Anschließend wird der Name der Instanz angegeben, wie die nachfolgende Abbildung59 darstellt.

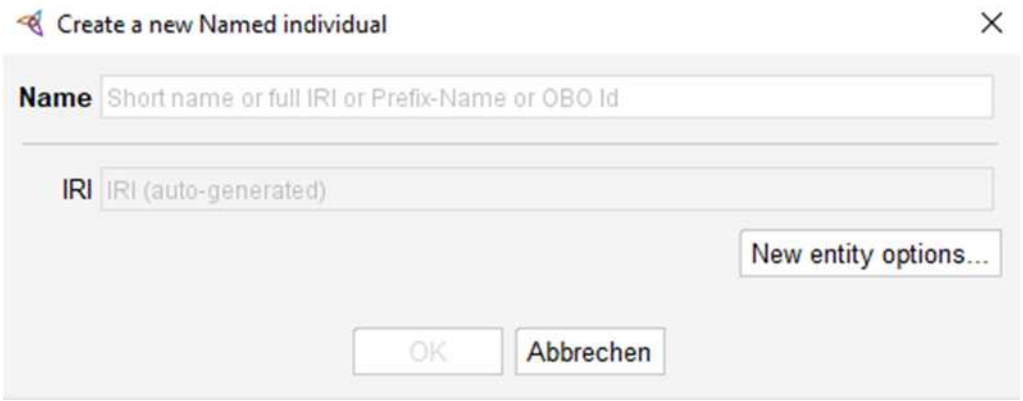


Abbildung 59: Name der Instanz

Sobald der Name der Instanz – hier für die Instanz *RisikoIdentifizieren* – angelegt wurde, ist die konstruierte Instanz in der Auflistung aller Instanzen zu finden und kann einer Klasse zugeordnet werden. Siehe dazu die nachfolgende Abbildung60.

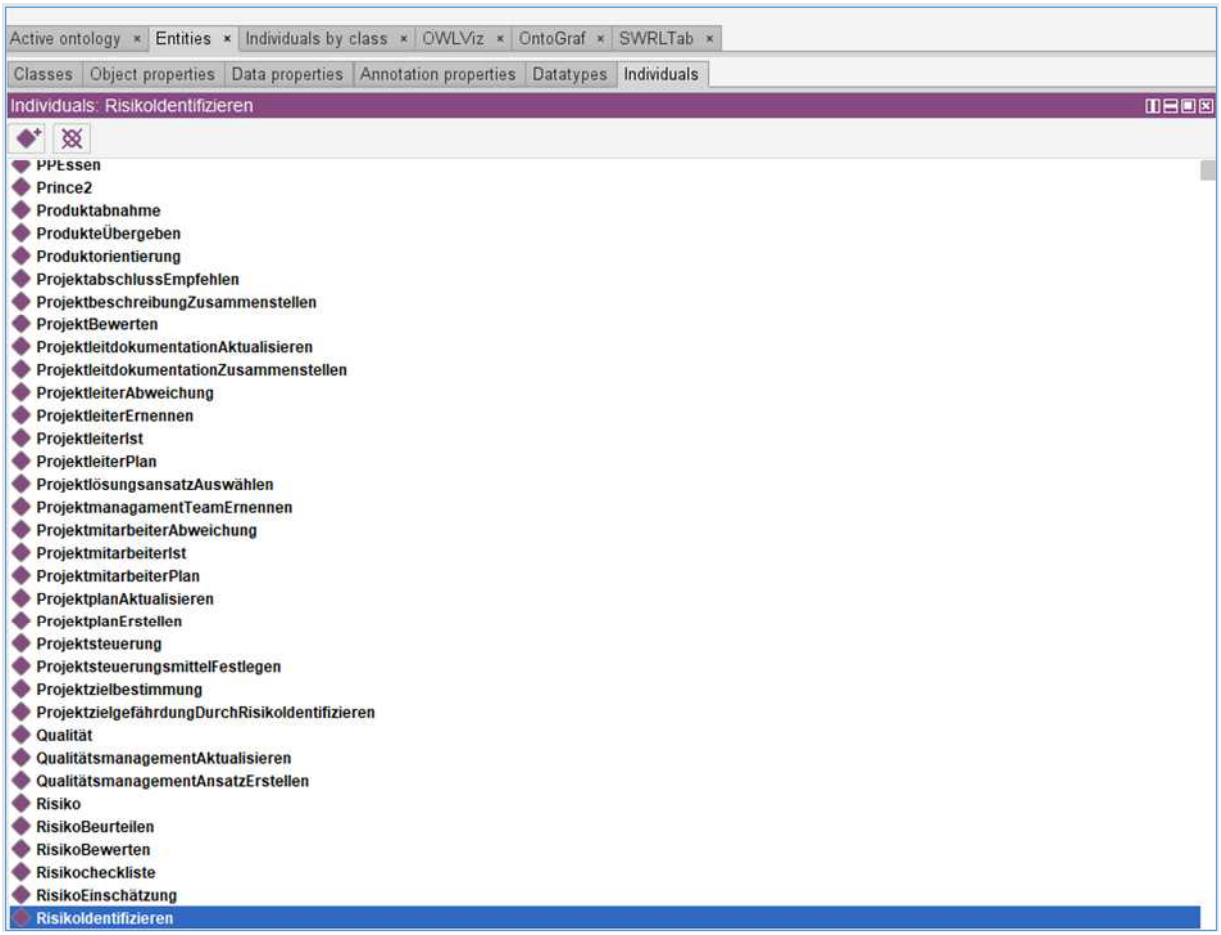


Abbildung 60: Anzeige der neu angelegten Instanz *RisikoIdentifizieren* in der Übersicht

Für die Zuordnung der Instanz *RisikoIdentifizieren* zu einer Klasse wird im Bereich „Types“ des Class Expression Editors geöffnet und die relevante Klasse für die Instanz ausgewählt.

Abbildung 61: Zuweisung einer Klasse *für die* Instanz *RisikoIdentifizieren* im Class Expression Editor

Im hier betrachteten Beispiel wird für die Instanz *RisikoIdentifizieren* die Klasse *RisikomanagementProzess* ausgewählt. Somit ist die Instanz *RisikoIdentifizieren* eine Instanz der Klasse *RisikomanagementProzess*, wie die nachfolgende Abbildung62 zeigt.

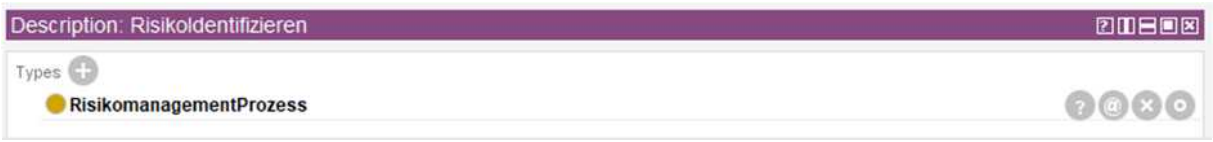


Abbildung 62: Zuweisung der Klasse *RisikomanagementProzess* zu der Instanz *RisikoIdentifizieren*

Nach der Zuordnung der Klasse *RisikomanagementProzess* müssen der Instanz *RisikoIdentifizieren* ihre Eigenschaften zugewiesen werden. Die Klasse *RisikomanagementProzess* hat eine nicht-taxonomische Relation *beinhaltetMaßnahme* mit der Kardinalität „*min* 1 *Maßnahme*“, sodass mindestens eine Maßnahme zugeordnet werden muss. Der Instanz *RisikoIdentifizieren* werden die Maßnahmen *KontextIdentifizieren* und *GefährdungsIdentifizieren* zugeordnet, wie in der nachfolgenden Abbildung63 dargestellt ist.

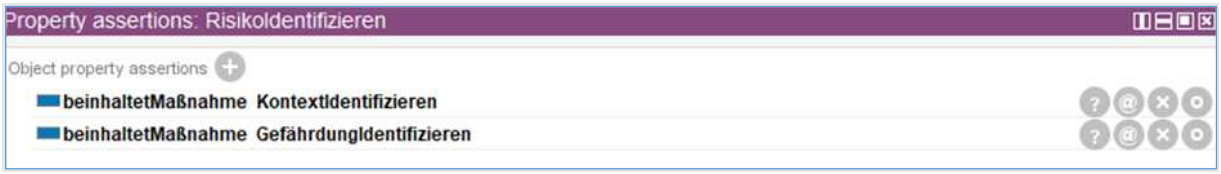


Abbildung 63: Zugewiesene Eigenschaften zu der Instanz *RisikoIdentifizieren*

Die Maßnahmen *KontextIdentifizieren* und *GefährdungIdentifizieren* sind ebenfalls Instanzen. Sie gehören der Klasse *Maßnahme* an.

Die nachfolgende Abbildung64 illustriert zusammenfassend die vorgenommene Konstruktion der Instanz *RisikoIdentifizieren*. Die gestrichelte Linie veranschaulicht die nicht-taxonomische Beziehung auf der Instanzenebene, die bereits durch die durchgezogene Linie auf der Klassenebene dargestellt wird.

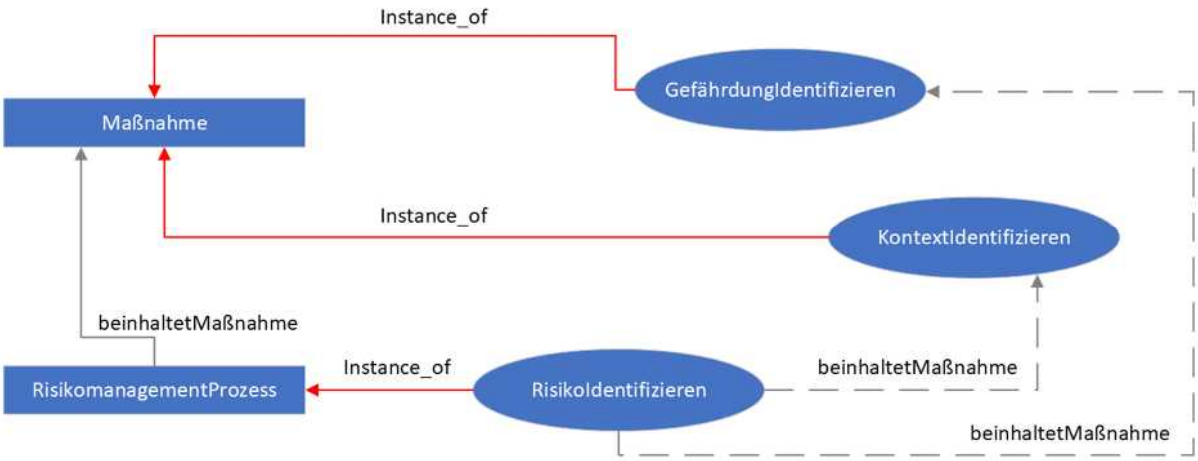


Abbildung 64: Instanzen *RisikoIdentifizieren*, *GefährdungIdentifizieren* und *KontextIdentifizieren* im Kontext ihrer Klassen

Im Folgenden werden die Instanzen *PRINCE2* und *RisikomanagementPRINCE2* hinsichtlich ihrer Konstruktion exemplarisch erläutert.

Die nachfolgende Tabelle52 beinhaltet die Instanz *PRINCE2*, die zur Klasse *Projektmanagementmethode* gehört.

Instanz: PRINCE2	
Klasse: Projektmanagementmethode	
Eigenschaft	**Wert**
bestehtAusThema	Änderung
bestehtAusThema	Risiko
bestehtAusThema	Qualität
bestehtAusThema	BusinessCase
bestehtAusThema	Fortschritt
bestehtAusThema	Organisation
bestehtAusThema	Plan
bestehtAusGrundprinzip	LernenAusErfahrung
bestehtAusGrundprinzip	SteuernNachDemAusnahmeprinzip
bestehtAusGrundprinzip	FortlaufendeGeschäftlicheRechtftigung
bestehtAusGrundprinzip	ProduktOrientierung
bestehtAusGrundprinzip	AnpassendieProjektumgebung
bestehtAusGrundprinzip	SteuerungÜberManagementphase
bestehtAusGrundprinzip	DefinierteRolleUndVerantwortlichkeit
bestehtAusProzess	SteuernEinerPhase
bestehtAusProzess	VorbereitenEinesProjektes
bestehtAusProzess	InitierenEinesProjektes
bestehtAusProzess	AbschließenEinesProjektes
bestehtAusProzess	LenkenEinesProjektes
bestehtAusProzess	ManagenEinesPhasenübergangs
bestehtAusProzess	AbschließenEinesProjektes

Tabelle 52: Darstellung der Instanz *PRINCE2*

Die Instanz *PRINCE2* besitzt die Eigenschaften *bestehtAusThema*, *bestehtAusGrundprinzip* und *bestehtAusProzess*, und zwar jeweils mit den Kardinalitäten „exactly 7“. Die Instanz *PRINCE2* kann auf diese Weise mit den sieben Grundprinzipien, den sieben Themen und den sieben Prozessen aus der Projektmanagementmethode (dem „Framework“) PRINCE2 – vgl. AXELOS (2015), 34-36, 47-59 u. 135-140 – in Verbindung gebracht werden.

Die nachfolgende Abbildung65 verdeutlicht beispielhaft die automatisierte Konstruktion der globalen Instanz *PRINCE2* in einem ontologiegestützten CBR-System, das mithilfe des CBR-Tools jCORA konstruiert wurde. Diese Instanz ist über die nicht-taxonomische Relation *bestehtAusProzess* mit der Instanz *AbschließenEinesProjektes* verknüpft. Letztere ist wiederum über die nicht-taxonomische Relation *beinhaltetMaßnahme* mit anderen Instanzen verbunden, beispielsweise mit der Instanz *Produktabnahme*. Auf diese Weise können die in PRINCE2 definierten Maßnahmen des Prozesses „Abschließen eines Projektes“ ausgedrückt werden.

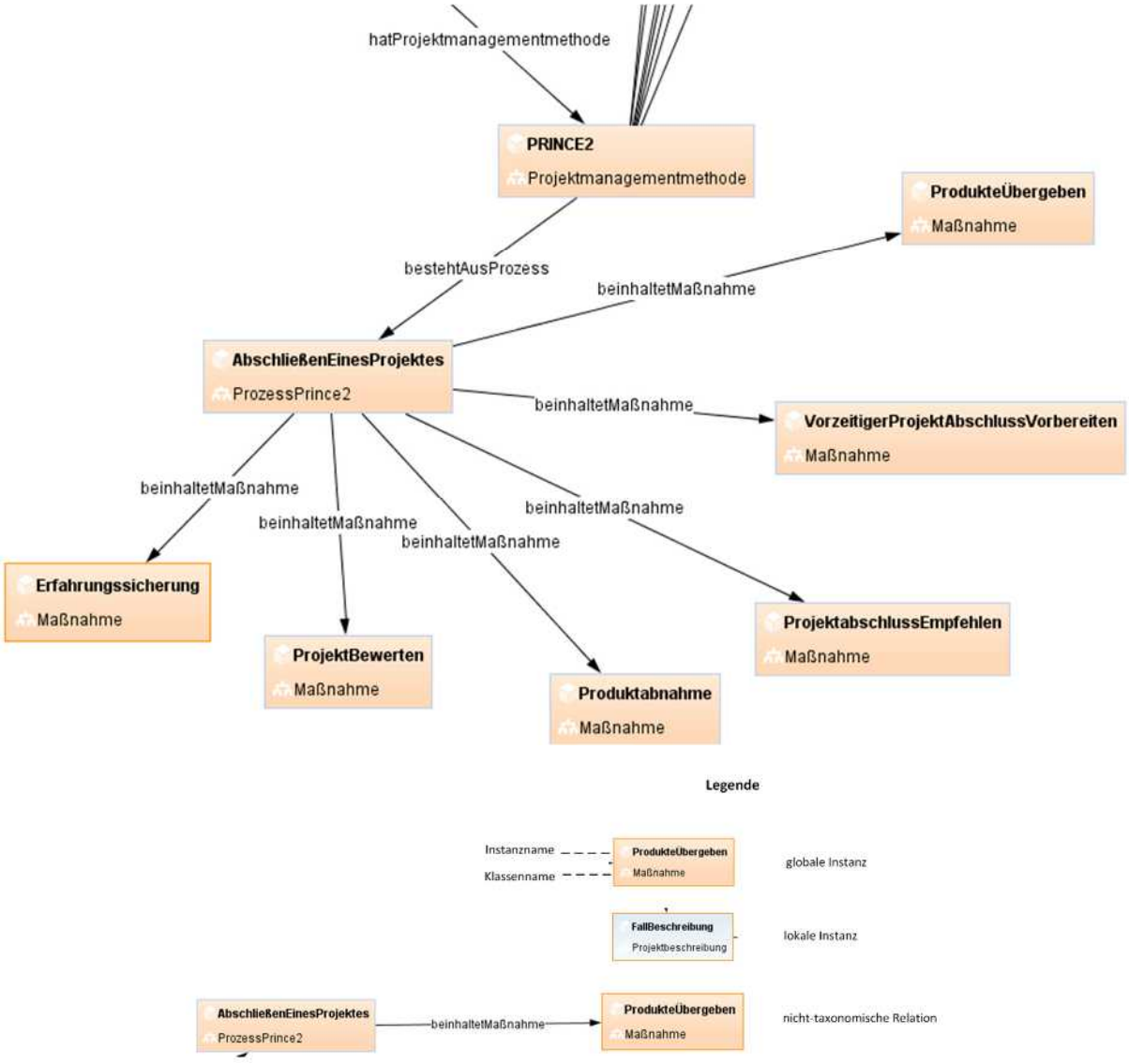


Abbildung 65: Die Exemplarische Darstellung von globalen Instanzen am Beispiel der Instanz *PRINCE2* in jCORA

Die nachfolgende Tabelle53 zeigt die Instanz *RisikomanagementPRINCE2* mit den zugehörigen Eigenschaften.

Instanz: RisikomanagementPRINCE2	
Klasse: Risikomanagement	
Eigenschaft	**Wert**
bestehtAusRisikomanagementProzess	RisikoBewerten
bestehtAusRisikomanagementProzess	RisikoMaßnahmenImplementieren
bestehtAusRisikomanagementProzess	RisikoIdentifzieren
bestehtAusRisikomanagementProzess	RisikoPlanen

Tabelle 53: Darstellung der Instanz *RisikomanagementPRINCE2*

Die Instanz *RisikomanagementPRINCE2* besitzt als Eigenschaft die nicht-taxonomische Relation *bestehtAusRisikomanagementProzess* mit der Kardinalität „exactly 4“ für die Klasse *RisikomanagementProzess* im Nachbereich der Relation. Die Instanz *RisikomanagementPRINCE2* ist nach dem Risikomanagement von PRINCE2 konstruiert, welche als Eigenschaftswerte folgende Instanzen der Klasse *RisikomanagementProzess* besitzt:

- *RisikoPlanen*
- *RisikoMaßnahmenImplementieren*
- *RisikoIdentifizieren*
- *RisikoBewerten*

Die Risikomanagementprozesse sind in PRINCE2 an den DEMING-Kreis angelehnt. Der DEMING-Kreis wird auch als PDCA-Zyklus, DEMING-Zyklus oder SHEWHART-Zyklus (oder auch „Rad“ oder „Kreis“) bezeichnet; vgl. beispielsweise KALS (2021), S. 287; ANGERMEIER (2016). Der DEMING-Zyklus beschreibt im Allgemeinen einen iterativen vierstufigen Prozess als Zyklus zur stetigen Verbesserung von Arbeitsprozessen und wurde durch den US-amerikanischen Statistiker DEMING eingeführt; vgl. SYSKA (2006), S. 100. Der DEMING-Zyklus ist in jeder Domäne anwendbar, wobei das Akronym PDCA für Plan, Do, Check und Act steht und die vier Prozesse des Zyklus beschreibt. Der zentrale Punkt des DEMING-Zyklus ist es, dass – unabhängig davon, wie die einzelnen Prozessschritte bezeichnet werden – die Auswirkungen eines Prozessschritts überprüft werden und diese Erkenntnisse wiederum rückwirkend Einfluss auf die Steuerung des gesamten Zyklus haben. Streng genommen besitzt der Risikomanagementprozess von PRINCE2 einen weiteren Prozessschritt, der nicht im DEMING-Zyklus enthalten ist. Dieser Prozessschritt, der als „Risiko kommunizieren“ bezeichnet wird, erfolgt durchgehend während des Zyklus; vgl. AXELOS (2018), S. 124. Aufgrund der Tatsache, dass der Prozessschritt „Risiko kommunizieren“ parallel zu allen Risikomanagementprozessen erfolgt, wird dieser Prozess im vorliegenden Beitrag als Maßnahme und nicht als separater Prozessschritt

definiert. Diese Konstruktionsentscheidung widerspricht nicht den Prozessschritten des Risikomanagements von PRINCE2, da in der Aufzählung der Prozessschritte in AXELOS (2018), S. 124, ebenfalls von vier nacheinander folgenden Prozessschritten gesprochen wird.

Die Instanz *RisikomanagementPRINCE2* wurde unter Verwendung der zugrunde gelegten PRINCE2- und Risikomanagement-Ontologie erstellt und anschließend für die sicherheitskritische IT-Projekt-Ontologie angepasst. Die Anpassungen basieren im Wesentlichen darauf, dass die zugrunde gelegte PRINCE2- und Risikomanagement-Ontologie für die Instanz *RisikomanagementPRINCE2* Eigenschaften vorsieht, die fälschlicherweise als Risikomanagementprozesse klassifiziert werden. Ein Beispiel hierfür ist die Instanz *RisikoregisterDokumentieren*, die in der ursprünglichen PRINCE2- und Risikomanagement-Ontologie als Risikomanagementprozess definiert ist, jedoch lediglich eine Maßnahme darstellt und als Eigenschaft für alle vier Risikomanagementprozesse vorgesehen werden sollte.

Die nachfolgende Abbildung66 stellt die angepasste Instanz *RisikomanagementPRINCE2* aus der sicherheitskritischen IT-Projekt-Ontologie dar, die sich an den von PRINCE2 definierten DEMING-Zyklus anlehnt. Jeder Risikomanagementprozess hat zusätzliche Maßnahmen als Eigenschaften. Die Abbildung66 zeigt dies beispielhaft für den Risikomanagementprozess *RisikoIdentifizieren*, der die Maßnahmen *KontextIdentifizieren* und *Gefährdung Identifizieren* beinhaltet.

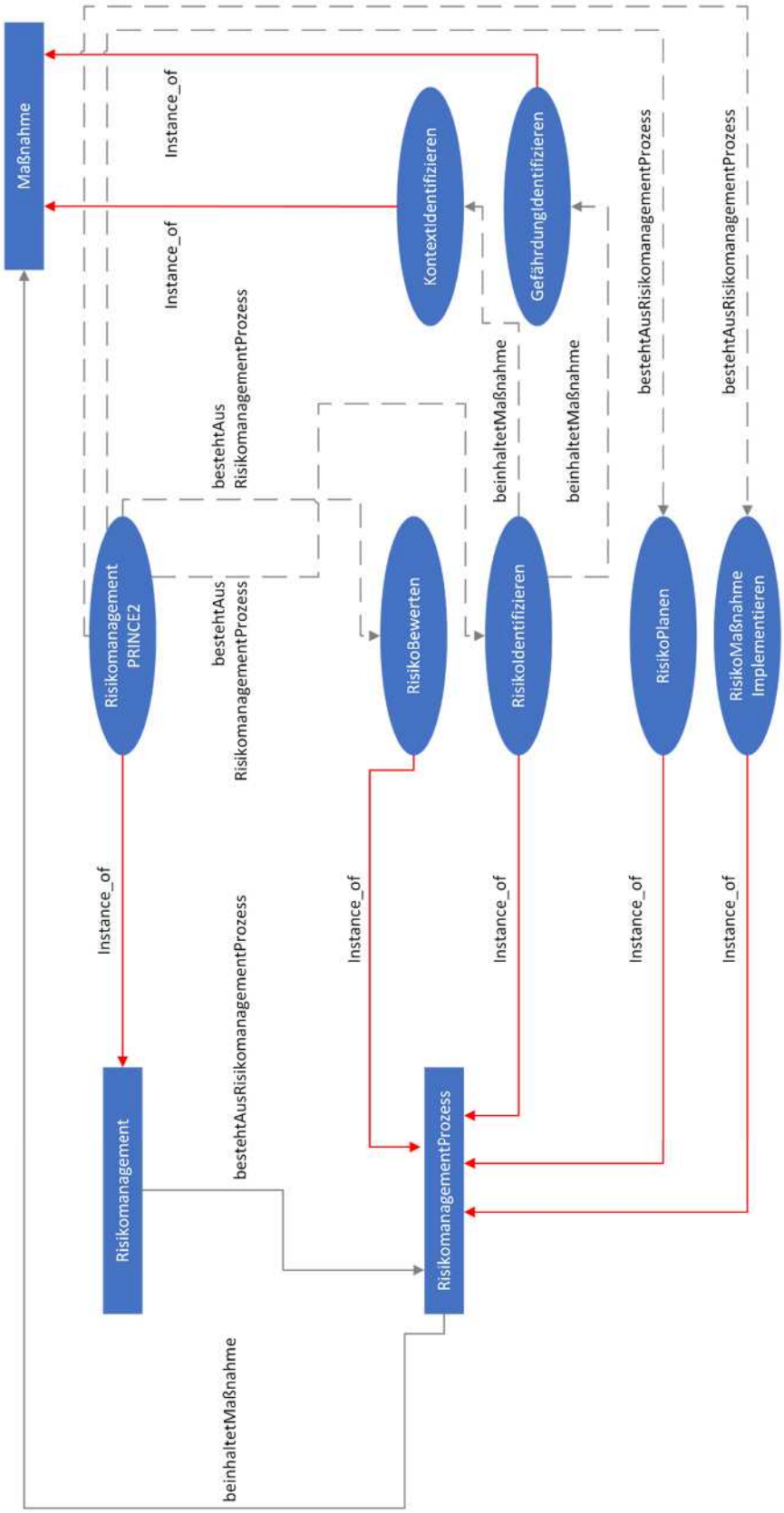


Abbildung 66: Risikomanagement von PRINCE2
mit den dazugehörigen Risikomanagementprozessen

Durch die Verwendung der Instanz *RisikomanagementPRINCE2* aus der sicherheitskritischen IT-Projekt-Ontologie können die Risikomanagementprozesse von PRINCE2 automatisiert im CBR-Tool jCORA konstruiert werden, wie die nachfolgende Abbildung67 exemplarisch darstellt.

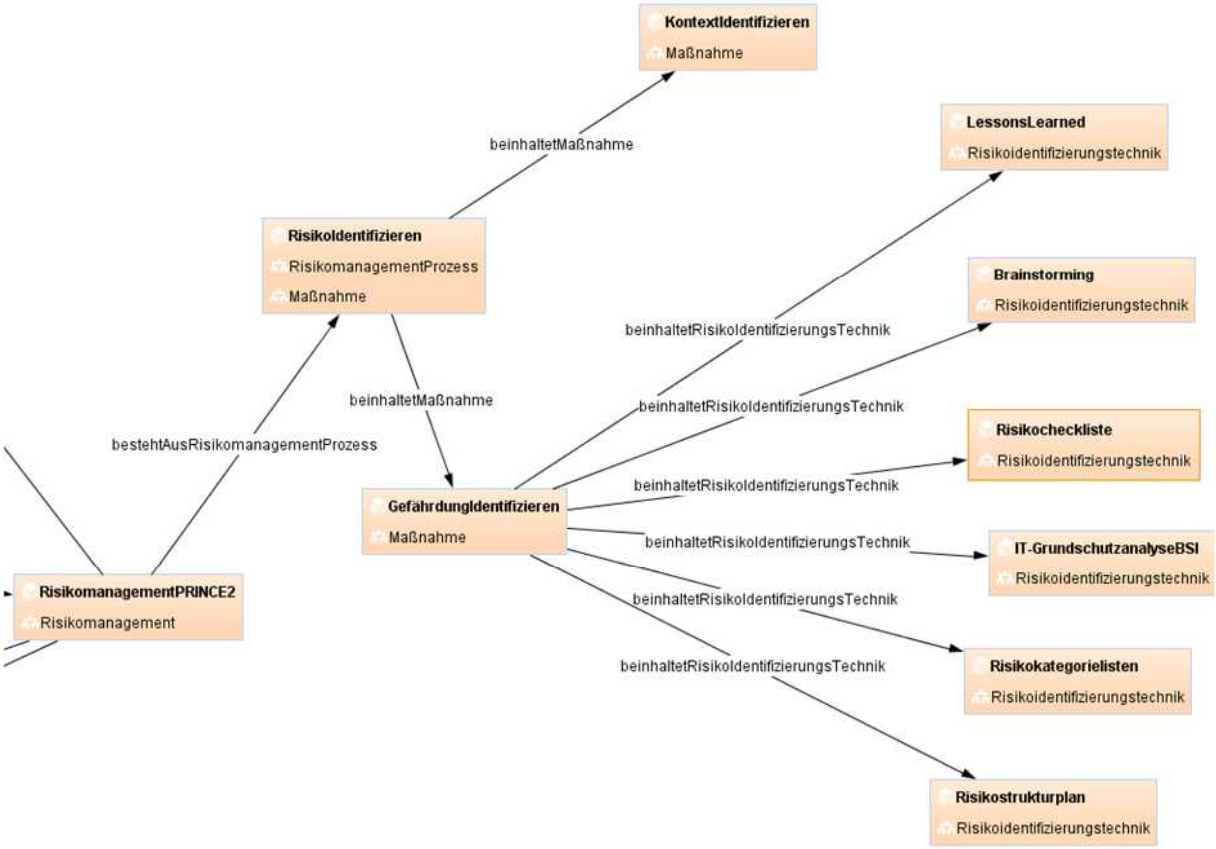


Abbildung 67: Darstellung von globalen Instanzen am Beispiel der Instanz *RisikomanagementPRINCE2* in jCORA

In der Abbildung67 lässt sich erkennen, dass die Instanz *RisikoIdentifizieren* (erste Zeile in ihrem ontologiezugehörigen Knoten in Fettdruck) zu zwei Klassen gehört (Polymorphismus), nämlich sowohl zur Klasse *Risikomanagementprozess* als auch zur Klasse *Maßnahme* (zweite und dritte Zeile in ihrem ontologiezugehörigen Knoten in Normaldruck).

3.3 Case-based Reasoning auf der Grundlage einer sicherheitskritischen IT-Projekt-Ontologie

3.3.1 Ontologiegestütztes Case-based-Reasoning-System mithilfe von jCORA

3.3.1.1 Beschreibung des prototypischen CBR-Tools jCORA

Das CBR-Tool jCORA („java based Case- and Ontology-based Reasoning Application“) ist eine prototypische Software für die Konstruktion von ontologiegestützten CBR-Systemen. Sie wurde für die „intelligente“ Wiederverwendung von Erfahrungswissen im betrieblichen Projektmanagement entwickelt. Dieses CBR-Tool wurde am Institut für Produktion und Industrielles Produktionsmanagement (PIM) der Universität Duisburg-Essen im Rahmen des BMBF-Verbundprojekts OrGoLo konzipiert sowie implementiert und im Zuge des BMBF-Verbundprojekts KI-LiveS weiterentwickelt. Vgl. ZELEWSKI/HEEB/SCHAGEN (2022), S. 225-251; BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 475-541.

Eine wesentliche Voraussetzung für die Anwendung des CBR-Tools jCORA ist das Vorhandensein einer Ontologie, in diesem Fall der sicherheitskritischen IT-Projekt-Ontologie. Aufgrund der Erforderlichkeit einer Ontologie wird bei jCORA von einem ontologiegestützten CBR-Tool gesprochen. In jCORA werden Ontologien in den Einstellungen als lokaler Pfad zu einer OWL-Datei angegeben. Ontologien im OWL-Format werden als Austauschformat für Ontologien zwischen einem Ontologieeditor und einem ontologiegestützten CBR-System benötigt. Dies wird in der nachfolgenden Abbildung 68 veranschaulicht.

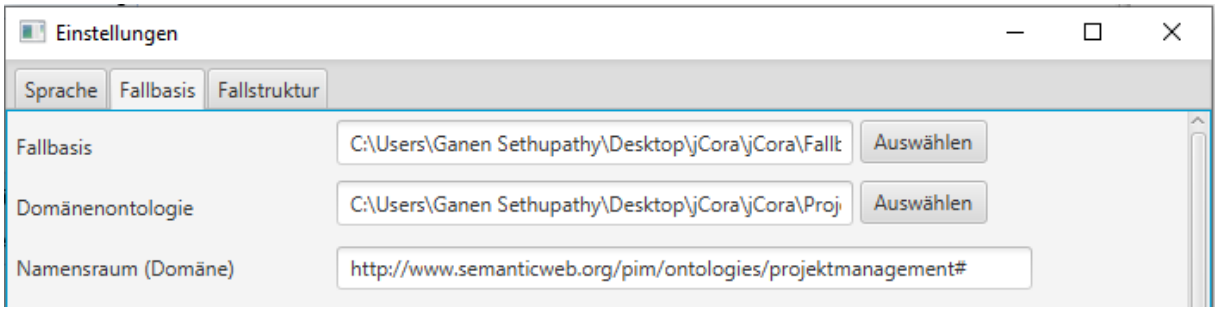


Abbildung 68: Einstellungen im CBR-Tool jCORA

In der Ontologie werden diejenigen sprachlichen Ausdrucksmittel spezifiziert, die als notwendig oder zumindest als hilfreich erachtet werden, um im vorliegenden Fall das Erfahrungswissen aus sicherheitskritischen IT-Projekten in jCORA zu repräsentieren.

Der Arbeitsbereich von jCORA liegt im Bereich der lokalen, projektspezifischen Instanzen und ihrer Instanzwerte. Der Ontologieeditor Protégé beschränkt sich dagegen auf die zugrunde liegende Ontologie und agiert überwiegend auf der Klassenebene einschließlich der taxonomischen Relation „is_a“ und weiterer nicht-taxonomischer Relationen zwischen Instanzen. Hinzu kommen globale Instanzen, die projektübergreifend für die gesamte Ontologie gelten, sowie

unter Umständen auch Regeln (des Öfteren auch als „Axiome“ bezeichnet), wie vor allem die bereits vorgestellten Semantic Web Rules.

Die Grenze zwischen der Klassen- und der Instanzenebene lässt sich nicht immer einfach vornehmen. Dies begründet sich darin, dass sowohl in einem Ontologieeditor wie Protégé als auch in einem CBR-Tool wie jCORA Instanzen konstruiert werden können. Wie bereits erläutert, beziehen sich *globale Instanzen*, die mit Protégé konstruiert werden, nicht ausschließlich auf einen einzelnen, projektspezifischen Realitätsausschnitt, sondern stellen sprachliche Ausdrucksmittel projektübergreifend bereit, mit denen sich Instanzen aus beliebigen projektbezogenen Realitätsausschnitten modellieren lassen, beispielsweise projektübergreifende Managementmethoden wie PRINCE2. In einigen Quellen wird bei den in einer Ontologie spezifizierten Instanzen von einer globalen Wissenskomponente gesprochen. Im Gegensatz dazu beziehen sich *lokale Instanzen* als lokale Wissenskomponenten ausschließlich auf ein einzelnes Projekt. Dieser Unterschied wird auch durch die verschiedenen Anwendungsbereiche des CBR-Tools jCORA und des Ontologie-Editors Protégé deutlich: Das CBR-Tool jCORA arbeitet auf der Ebene der lokalen, projektspezifischen Instanzen und ihrer zugehörigen Werte, während sich der Ontologie-Editor Protégé auf die zugrunde liegende Ontologie konzentriert und hauptsächlich auf der Klassenebene einschließlich globaler, projektübergreifender Instanzen agiert. Die Frage, ob eine Instanz global oder lokal konstruiert wird, lässt sich nur kontextabhängig beantworten.

Darüber hinaus kann die Unterscheidung zwischen globalen und lokalen Instanzen Auswirkungen auf den Algorithmus zur Ähnlichkeitsberechnung im CBR-Tool jCORA haben; vgl. SETHUPATHY (2024), S. 324; ZELEWSKI/SCHAGEN (2022), S. 32. Dort wird gezeigt, dass globale Instanzen zu fehlerhaften Ähnlichkeitsberechnungen in jCORA führen können, wenn diese globalen Instanzen trotz ihres Charakters als globale Wissenskomponenten bei der Spezifizierung von Projekten (Fällen)[3] in jCORA um lokale Relations- oder Attributswerte erweitert werden. Dies stellt eine erhebliche Funktionseinschränkung aufgrund einer Korrektheitsunsicherheit bei der praktischen Verwendung des CBR-Tools jCORA dar.

Die *Fallstruktur* besitzt in einem ontologiegestützten CBR-System eine herausragende Bedeutung. Laut ASSALI/LENNE/DEBRAY (2010), S. 105; WATSON (2003), S. 27-28; WATSON (1998), S. 176-177, ist bezüglich der Fallstrukturen von CBR-Tools zwischen einer homogenen und einer heterogenen Fallstruktur zu unterscheiden. Wenn jeder betrachtete Fall in einem CBR-Tool die gleichen Klassen, Relationen und Attribute besitzt, dann wird von einer homogenen Fallstruktur gesprochen. Andernfalls liegt eine heterogene Fallstruktur vor. Vgl. BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 491.

3) In diesem Beitrag werden die Bezeichnungen „Fälle“ (Cases) und „Projekte“ synonym verwendet, weil das *fallbezogene* Case-based Reasoning ausschließlich für das Management der Wiederverwendung von *projektbezogenem* Erfahrungswissen eingesetzt wird, sodass zwischen Fällen des Case-based Reasonings im Allgemeinen und Projekten im Sinne des (computergestützten) Wissensmanagements für Projekte nicht unterschieden wird. In diesem Beitrag werden die Bezeichnungen „Fälle“ versus „Projekte“ so verwendet, wie es im jeweils aktuellen Argumentationskontext plausibel erscheint.

Diverse Fachliteratur setzt für die Anwendung von Case-based Reasoning voraus, dass die Fallstruktur ex ante bekannt ist, für alle Fälle übereinstimmt und sich im Laufe der Nutzung eines CBR-Tools nicht verändert. Die Fallstruktur gilt in diesem Fall als homogen. Dadurch lassen sich Fälle hinsichtlich ihrer Eigenschaften besser miteinander vergleichen und Ähnlichkeiten zwischen Fällen können mittels einfacher Heuristiken berechnet werden; vgl. ASSALI/LENNE/DEBRAY (2010), S. 98. In komplementärer Weise wird betont, dass CBR-Tools mit einer heterogenen Fallstruktur als schwieriger umsetzbar gelten als CBR-Tools mit einer homogenen Fallstruktur; vgl. AVESANI/SUSI (2010), S. 183, ABOU ASSALI et al. (2009), S. 564. Die besondere Herausforderung heterogener Fallstrukturen wird im Rahmen der Ähnlichkeitsermittlung gesehen; vgl. ASSALI/LENNE/DEBRAY (2010), S. 98. Diese Einschätzung führte zur Entwicklung des CBR-Tools „jCORA“, das eine möglichst flexible Handhabung von heterogenen Fallstrukturen ermöglichen soll; vgl. BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 479. Auch die reale Komplexität von sicherheitskritischen IT-Projekten in der betrieblichen Praxis lässt vermuten, dass Projekte eine heterogene Fallstruktur (Projektstruktur) aufweisen.

Die *Wissensstrukturierung* in CBR-Tool jCORA untergliedert sich in drei wesentliche *projektbezogene* Komponenten. Es wird unterschieden zwischen:

- Projektbeschreibung,
- Projektlösung und
- Projektbewertung.

Die Funktionalität des CBR-Tools jCORA orientiert sich an dem früher beschriebenen CBR-Zyklus. Der funktionale Fokus von jCORA liegt auf der *Retrieve-Phase* des CBR-Zyklus. In der Retrieve-Phase wird mittels eines Ähnlichkeitsalgorithmus die *Ähnlichkeit* zwischen der Beschreibung eines neuen Projekts und den in der Projektwissensbasis gespeicherten Beschreibungen alter, bereits durchgeführter Projekte ermittelt. Der Ähnlichkeitsalgorithmus des CBR-Tools jCORA lehnt sich in seinen Grundzügen an den Ähnlichkeitsalgorithmus von BEIßEL (2011), S. 159-215, an. Der Ähnlichkeitsalgorithmus von BEIßEL weist jedoch verschiedene Mängel auf, die seine Praxistauglichkeit einschränken; vgl. BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 479-480. Dazu gehören vor allem folgende Einschränkungen:

- Der Ähnlichkeitsalgorithmus unterstützt keine rekursive Ablaufstruktur.
- Der Ähnlichkeitsalgorithmus weist Limitationen hinsichtlich der zulässigen Kardinalitäten von nicht-taxonomischen Relationen und Attributen auf.
- Der Ähnlichkeitsalgorithmus unterstützt keine Restriktionen, Konjunktionen und Disjunktionen sowie keine nicht-taxonomischen Relationen mit mehreren Werten im Nachbereich.

Im Rahmen der Ähnlichkeitsberechnung von zwei Projekten können im CBR-Tool jCORA Gewichte für Projektmerkmale, die von Benutzern als ähnlichkeitsrelevant erachtet werden, hinterlegt werden (darauf wird später zurückgekommen).

Zwar unterstützt das CBR-Tool jCORA die Retain-Phase ohne Einschränkung und automatisiert. Aber jCORA leidet hinsichtlich der Reuse- und der Revise-Phase des CBR-Zyklus – wie auch die meisten anderen derzeit bekannten CBR-Tools – unter erheblichen Limitationen:

- Während der Reuse-Phase wird eine automatisierte Anpassung an die Projektlösung eines möglichst ähnlichen alten, bereits durchgeführten Projekts nicht unterstützt. Es ist ausschließlich eine Kopierfunktion als „Nulladaption“ vorgesehen, die keine effektive Anpassung der Projektlösungen von alten an neue Projekte ermöglicht.
- Die Revise-Phase wird durch jCORA überhaupt nicht unterstützt.

Die beiden voranstehenden Anmerkungen weisen auf erheblichen Weiterentwicklungsbedarf des CBR-Tools jCORA hin.

3.3.1.2 Nutzung des CBR-Tools jCORA zur Fallspezifizierung

Nachfolgend wird die Funktionsweise des CBR-Tools jCORA mittels Screenshots beispielhaft erläutert. Da in jCORA als Tool für das ontologiegestützte *Case*-based Reasoning die Verwendung der Bezeichnung „Fall“ für „Cases“ im Vordergrund steht, wird im Folgenden vornehmlich von *Fällen* – anstelle der zugrunde liegenden und synonym gemeinten Projekte – gesprochen. Dennoch ist inhaltlich jeweils die Spezifizierung eines *Projekts* gemeint, über das Wissen – vornehmlich natürlichsprachliches Erfahrungswissen – in der Projektwissensbank eines ontologiegestützten CBR-Systems vorgehalten und zur Planung, Steuerung und Kontrolle neuer Projekte wiederverwendet werden soll.

Um im CBR-Tool in jCORA auf der Instanzenebene einen neuen Fall anzulegen, wird auf der Startseite von jCORA auf das Symbol „+“ geklickt:

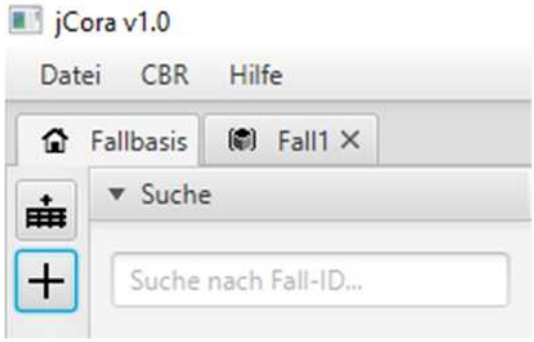


Abbildung 69: Anlegen eines neuen Falls

Anschließend kann für den Fall eine Fall-ID gewählt werden. In der nachfolgenden Abbildung70 wird dieser Schritt demonstriert und für den Fall die Bezeichnung „Kooperative_Leitstelle“ gewählt.

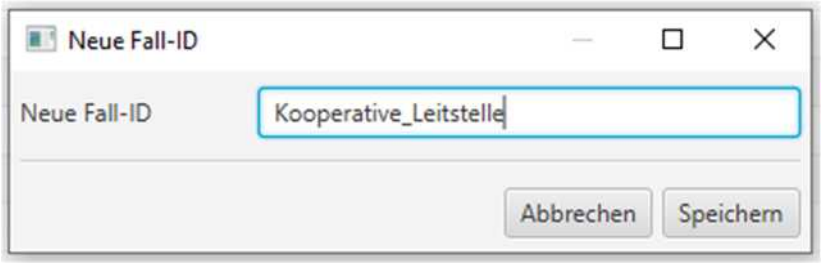


Abbildung 70: Das Anlegen einer Fall-ID

Sobald der Fall angelegt wurde, wird die bereits dargestellte dreiteilige Fallstruktur automatisch konstruiert und angezeigt; vgl. die nachfolgende Abbildung71.

Zur Falldarstellung dient jeweils ein *Fallgraph*. Jeder seiner *Knoten* führt zunächst in seiner ersten (oberen) Zeile in Fettdruck eine *Instanz* an, die zur Spezifizierung eines Falls auf der Instanzenebene dient. Zusätzlich steht in der zweiten (unteren) Zeile desselben Knotens in Normaldruck diejenige *Klasse*, zu der die Instanz des Knotens in der zugrunde liegenden Ontologie gehört. In der nachfolgenden Abbildung71 gehört beispielsweise die Instanz *Fall* zur Klasse *Projekt*, ebenso wie die Instanz *FallBeschreibung* zur Klasse *Projektbeschreibung* gehört.

Die gerichteten *Kanten* zwischen den Knoten eines Fallgraphs stellen jeweils eine *nicht-taxonomische Relation* dar, die jeweils als Kantenanschrift ausgewiesen wird und bezüglich derer gilt: Die beiden *Instanzen* aus dem Vorbereich der Relation (Beschriftung des Knotens am Kantenanfang in der ersten Zeile) und dem Nachbereich der Relation (Beschriftung des Knotens am Kantenende in der ersten Zeile) stellen als geordnetes Paar ein *Relationselement* der nicht-taxonomischen Relation dar, die als Kantenanschrift ausgewiesen wird. Darüber hinaus gilt aufgrund der zweifachen Beschriftung der Knoten des Fallgraphs, dass jede gerichtete Kante zwischen zwei Knoten eines Fallgraphs mit derjenigen nicht-taxonomischen *Relation* beschriftet ist, welche die beiden *Klassen* aus dem Vorbereich der Relation (Beschriftung des Knotens am Kantenanfang in der zweiten Zeile) und dem Nachbereich der Relation (Beschriftung des Knotens am Kantenende in der zweiten Zeile) miteinander verknüpft.

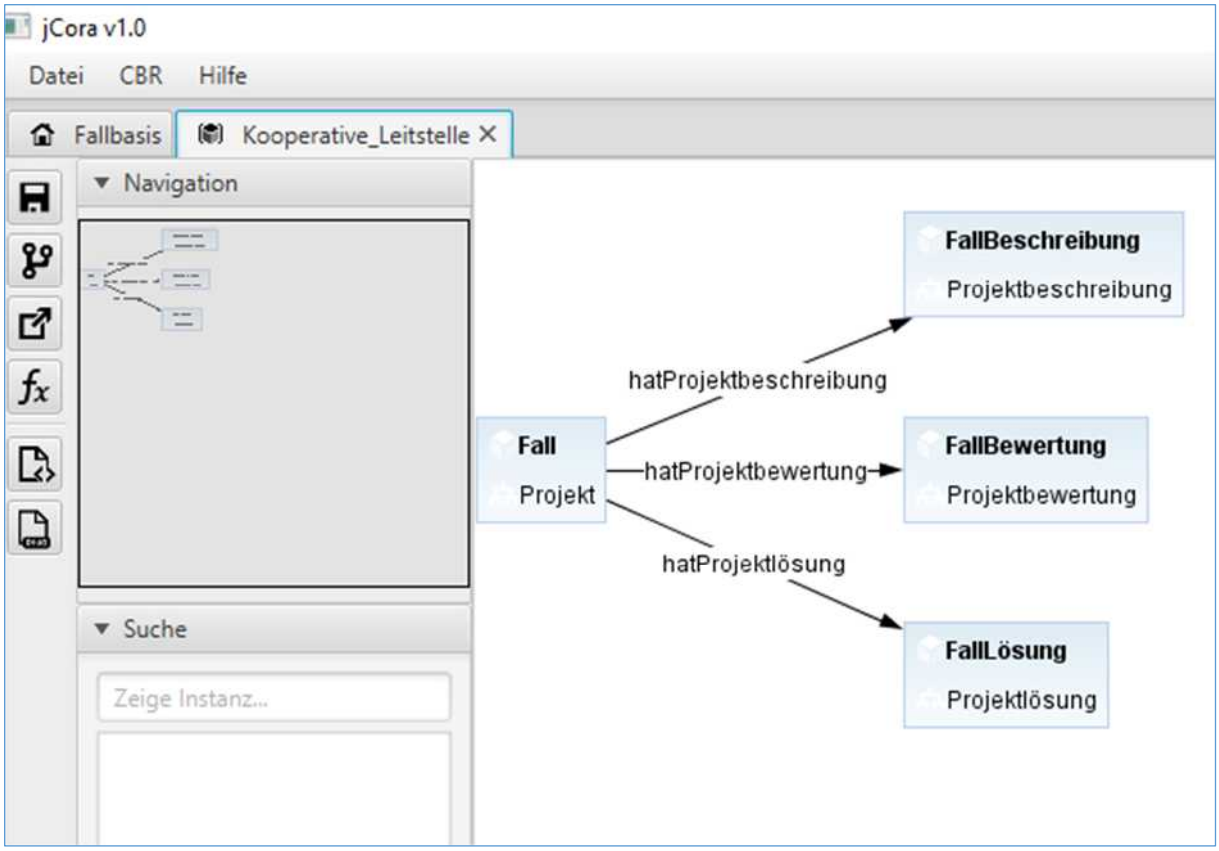


Abbildung 71: Ursprünglicher Fallgraph in jCORA

Die Instanz *Fall* stellt in jCORA eine Besonderheit dar. Da sie weder als globale Instanz in Protégé definiert wurde noch als konventionelle, fallspezifische Instanz in jCORA vorhanden ist, kann sie nicht in die Kategorisierung von globalen und lokalen Instanzen eingeordnet werden. Stattdessen wird diese Instanz durch die Einstellungen von jCORA definiert, wie in der nachfolgenden Abbildung72 dargestellt ist. Semantisch gesehen ergibt es keinen Sinn, eine Instanz *Fall* als Subklasse von *Projekt* zu betrachten, da ein Fall per se keine Subklasse eines Projekts ist, sondern die Begriffe „Projekt“ und „Fall“ synonym verwendet werden. Die Argumentation gilt analog auch für die Instanzen *FallBeschreibung*, *FallBewertung* und *FallLösung*.

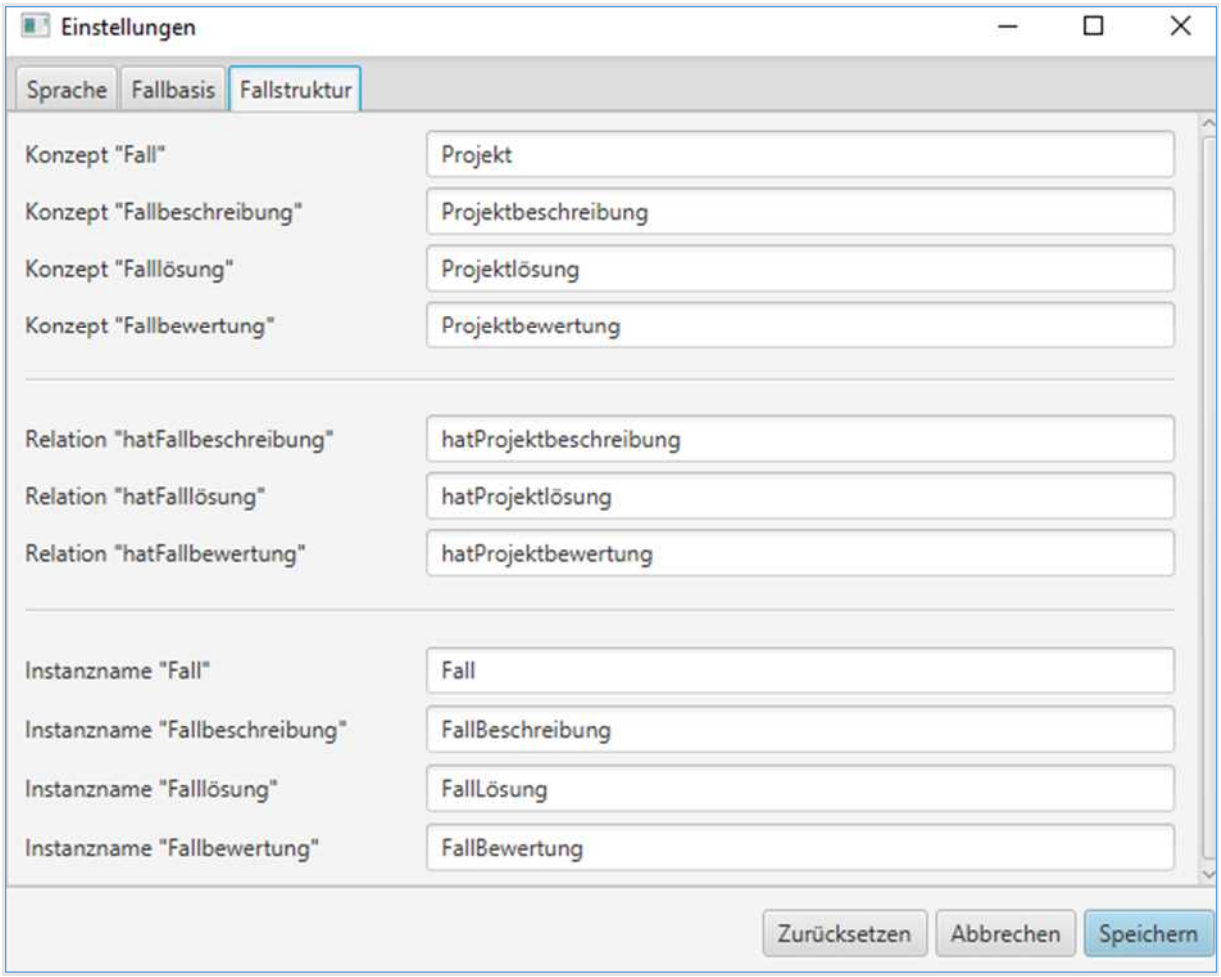


Abbildung 72: Einstellungen in jCORA

Um ein sicherheitskritisches IT-Projekt zu spezifizieren, wird ausgehend von der Instanz *Fall-Beschreibung* per Rechtsklick eine neue nicht-taxonomische Relation ausgewählt. Die nachfolgende Abbildung73 illustriert die Auswahl der möglichen nicht-taxonomischen Relationen, ausgehend von der Instanz *FallBeschreibung*. Hierbei werden nur die nicht-taxonomischen Relationen angezeigt, die in ihren Vorbereichen jeweils die Klasse *Projektbeschreibung* aufweisen.

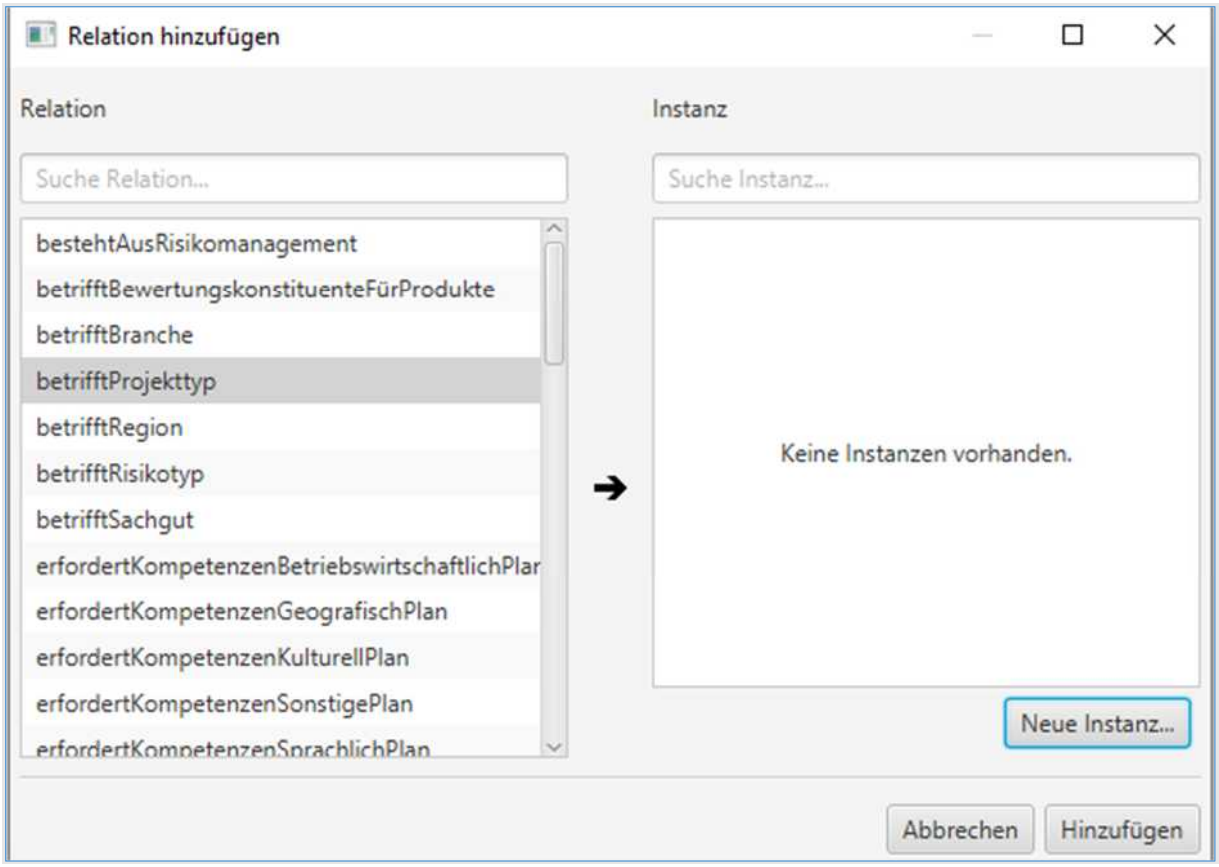


Abbildung73: Mögliche nicht-taxonomische Relationen
ausgehend von der Instanz *FallBeschreibung*

Im vorliegenden Kontext wird die nicht-taxonomische Relation *betrifftProjekttyp* ausgewählt, um eine Instanz zuzuordnen, die zur Klasse *SicherheitskritischesITProjekt* gehört, da diese im Nachbereich der nicht-taxonomischen Relation *betrifftProjekttyp* benötigt wird. Da bisher keine Instanzen der Klasse *SicherheitskritischesITProjekt* existieren, wird eine neue Instanz namens *Kooperative_Leitstelle* erstellt und der Klasse *SicherheitskritischesITProjekt* zugewiesen, wie in der nachfolgenden Abbildung74 veranschaulicht wird.

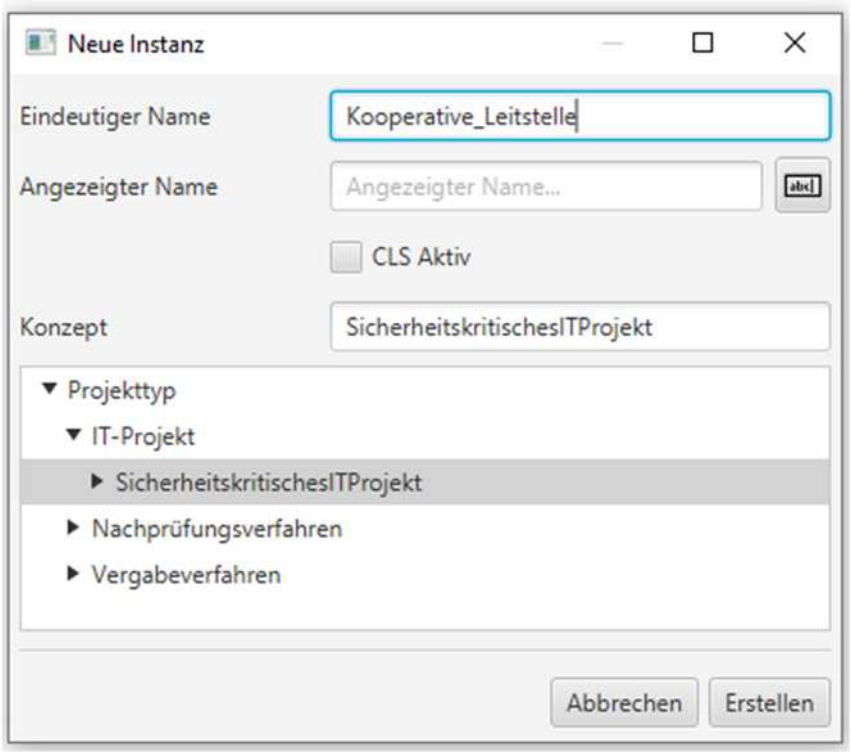


Abbildung 74: Erstellen der neuen Instanz *Kooperative_Leitstelle*

Nach der Konstruktion der Instanz *Kooperative_Leitstelle* kann diese im Nachbereich der nicht-taxonomischen Relation *betrifftProjekttyp* verwendet werden, wie in der nachfolgenden Abbildung75 dargestellt wird.

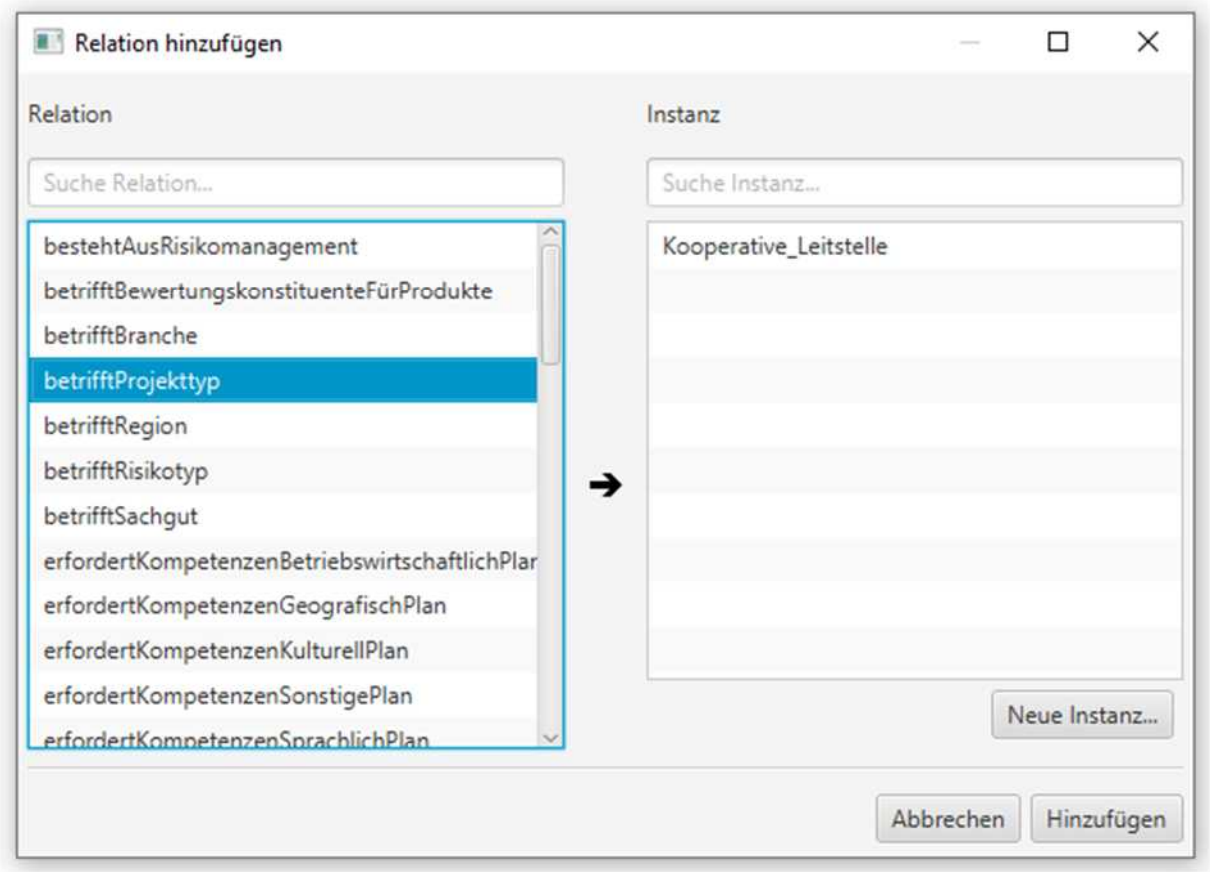


Abbildung 75: Auswahl der Instanz *Kooperative_Leitstelle*
als Element des Nachbereichs der nicht-taxonomischen Relation *betrifftProjekttyp*

Das hinzugefügte Relationselement der nicht-taxonomischen Relation *betrifftProjekttyp* wird im Anzeigefenster als Erweiterung des Fallgraphs um die Instanz *Kooperative_Leitstelle* aus der Klasse *SicherheitskritischesITProjekt* dargestellt.

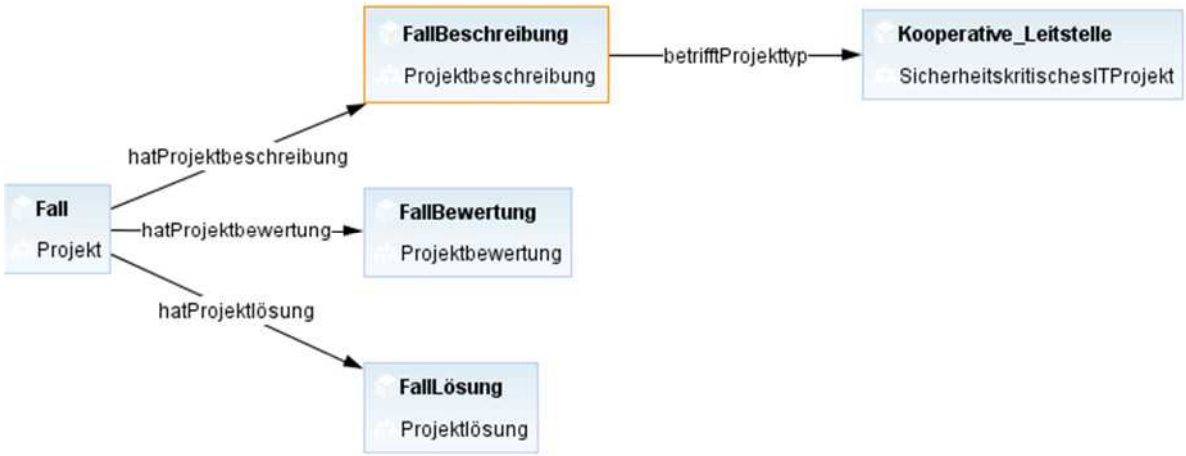


Abbildung 76: Erweiterter Fallgraph in jCORA

Im Folgenden werden der Instanz *Kooperative_Leitstelle* exemplarisch ausgewählte Eigenschaften eines sicherheitskritischen IT-Projekts zugeordnet. Die nachfolgende Abbildung77 veranschaulicht beispielhaft, wie der Fallgraph ausgehend von der Instanz *Kooperative_Leitstelle* durch nicht-taxonomische Relationen erweitert werden kann. Im Anzeigefenster werden nur diejenigen nicht-taxonomischen Relationen dargestellt, denen im Vorbereich die Klasse *SicherheitskritischesITProjekt* zugeordnet ist.

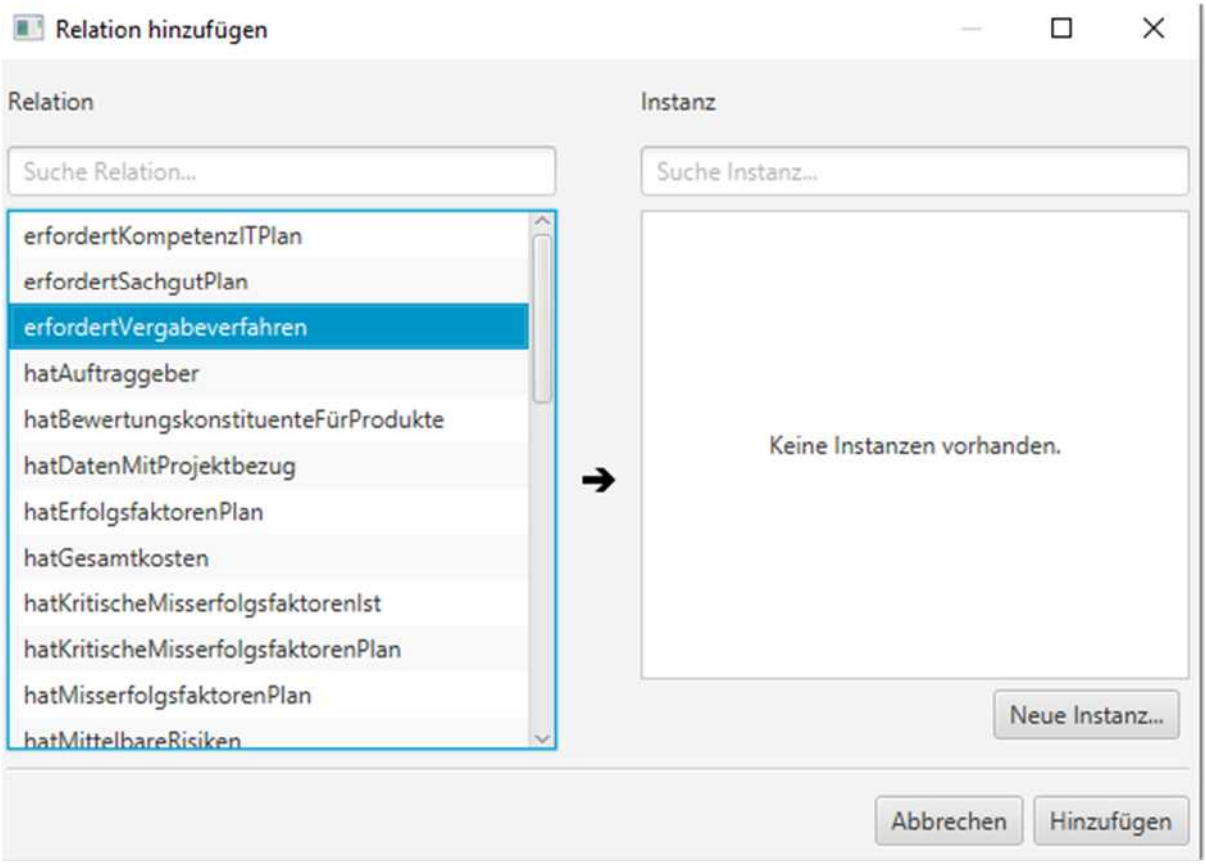


Abbildung 77: Anzeigefenster für das Hinzufügen einer nicht-taxonomischen Relation

Die nachfolgende Abbildung78 illustriert die Zuweisung der Attributswerte für die Attribute der Instanz *Kooperative_Leitstelle*. Hierzu wird das Symbol „+“ ausgewählt, um den Attributen konkrete Werte zuzuweisen. Das CBR-Tool jCORA ordnet dem ausgewählten Attribut automatisch denjenigen Datentyp zu, der in der zugrunde liegenden Ontologie für das jeweilige Attribut definiert wurde.

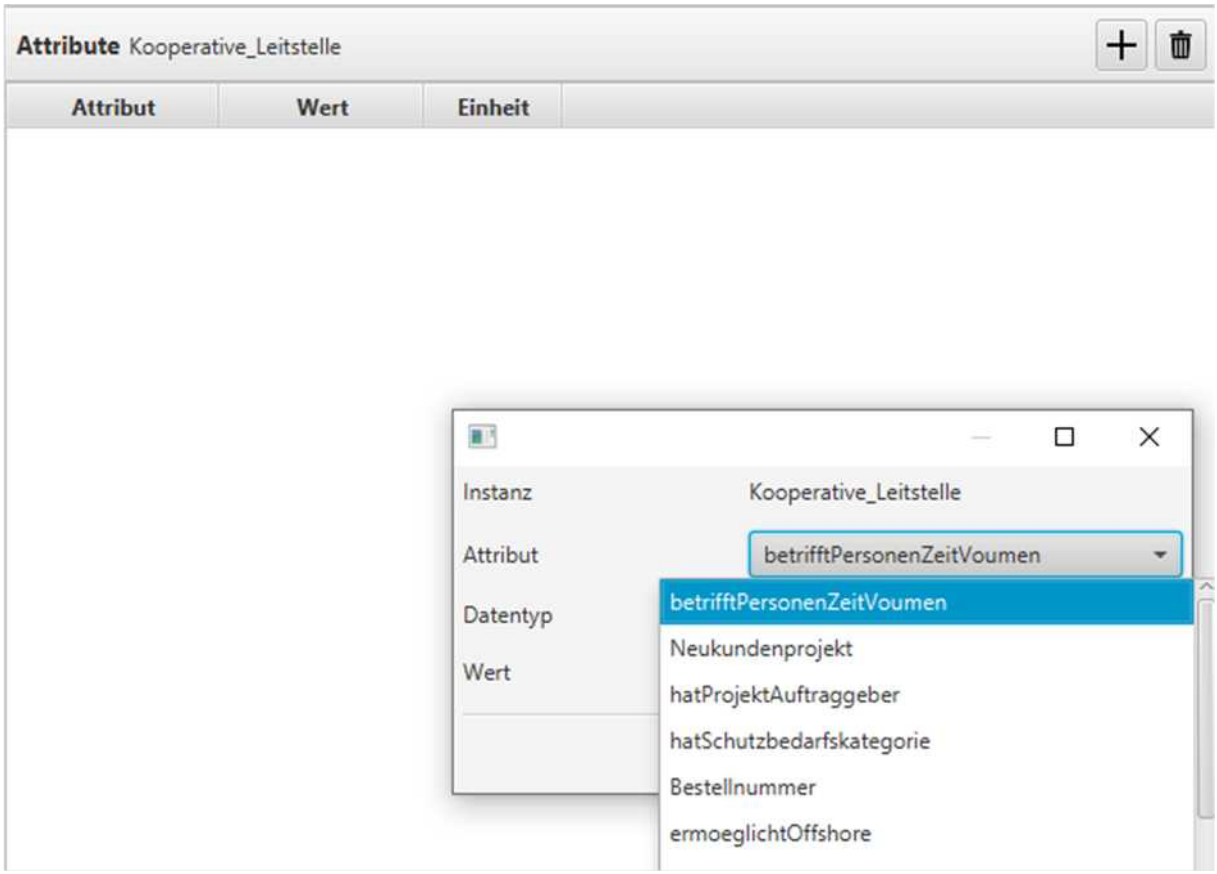


Abbildung 78: Zuweisung der Werte für die Attribute der Instanz *Kooperative_Leitstelle*

Die nachfolgende Abbildung 79 illustriert für die Instanz *Kooperative_Leitstelle* exemplarisch die Zuweisung der Werte für ihre Attribute.

Attribute Kooperative_Leitstelle

Attribut	Wert	Einheit
Projektname	Neuaufbau_einer_kooperativen_Leitstelle	(String)
Leistungsort	Haltern am See	(String)
hatSchutzbedarfsk...	Sehr Hoch	(String)
Landesrecht	Nordrhein-Westfalen	(String)
hatProjektablage	https://ablageOrt.InternSharepoint.de	(String)

Abbildung 79: Zugewiesene Werte für Attribute der Instanz *Kooperative_Leitstelle*

Auf diese Weise kann der Fallgraph von jCORA um Instanzen und zugehörige Relationselemente und Attributswerte erweitert werden, bis alle Wissenskomponenten, die für die Beschreibung des jeweils betrachteten sicherheitskritischen IT-Projekts als relevant erachtet werden, durch den Benutzer angelegt wurden.

Mit dem CBR-Tool jCORA lassen sich ontologiegestützte CBR-Systeme für komplexe sicherheitskritische IT-Projekte konstruieren, wie ein Ausschnitt in der nachfolgenden Abbildung80

beispielhaft zeigt. Obwohl in diesem Beitrag nur auf drei Fälle eingegangen wird, die eine deutlich geringere Komplexität als der in Abbildung80 auszugsweise dargestellte Fall aufweisen, wird durch die Abbildung80 verdeutlicht, dass mittels des CBR-Tools jCORA – trotz der Einschränkungen hinsichtlich seiner Anwendung – komplexe sicherheitskritische IT-Projekte als Fälle spezifiziert werden können. Weiterhin konnten durch die Fallkonstruktion fehlende Klassen, nicht-taxonomische Relationen und Attribute identifiziert und in Protégé nachmodelliert werden.

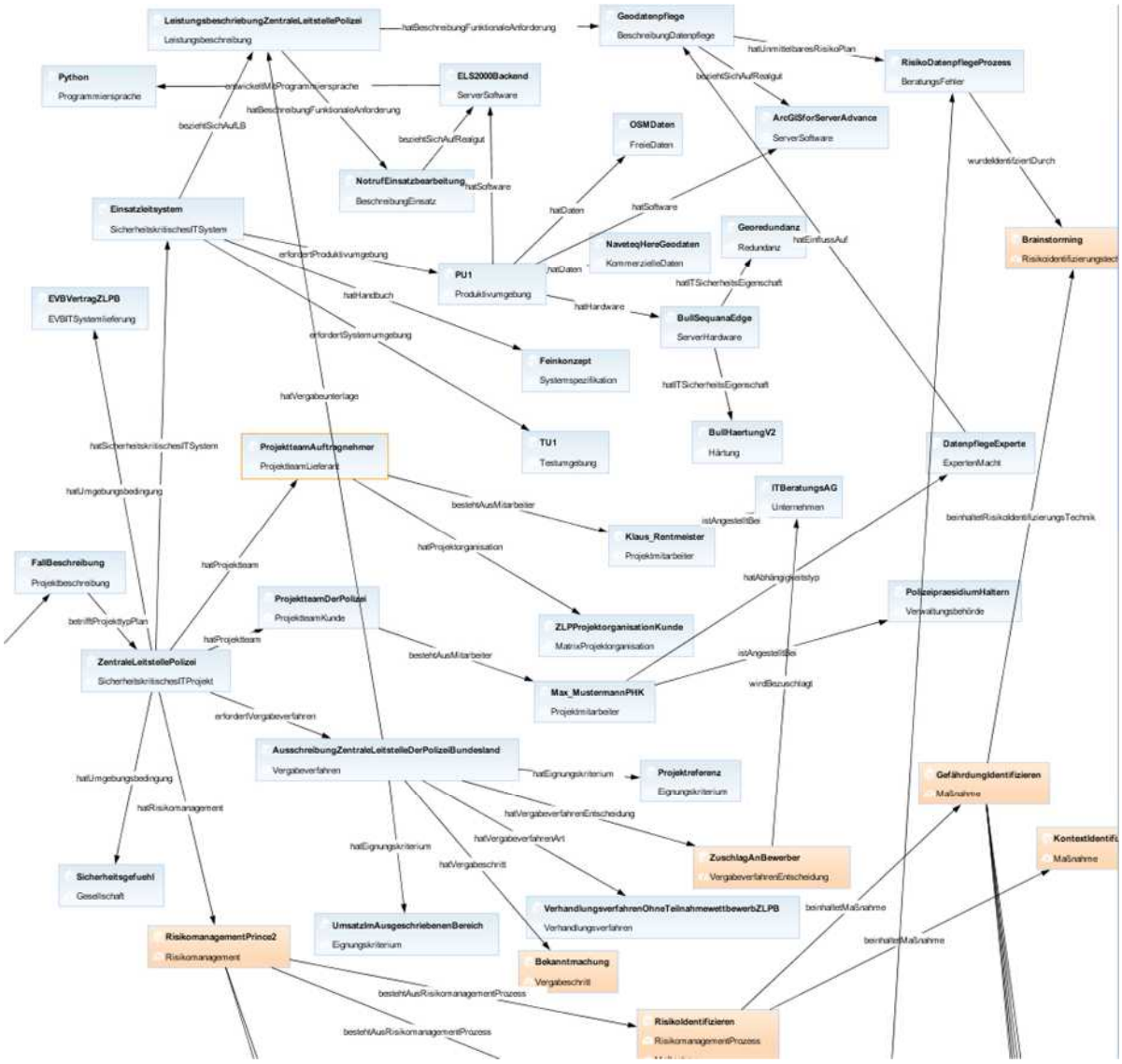


Abbildung 80: Auszug eines exemplarischen Falls aus jCORA

Die Abbildung80 zeigt beispielhaft die der Fallbeschreibung zugehörigen Wissenskomponenten. Das sicherheitskritische IT-Projekt „ZentraleLeitstellePolizei“ hat, wie in der Definition eines sicherheitskritischen IT-Projektes in diesem Beitrag gefordert, bereits ein Vergabeverfahren durchlaufen. Eignungskriterien des Vergabeverfahrens waren unter anderem die Projektre-

ferenz und der notwendige Umsatz im ausgeschriebenen Bereich. Im Rahmen dieses Vergabeverfahrens, bei dem es sich um ein Verhandlungsverfahren handelte, wurde die fiktive Firma ITBeratungAG beauftragt. Im Vergabeverfahren ist in der Leistungsbeschreibung eine Anforderung zur Geodatenpflege enthalten. Diese Geodatenpflege wird im sicherheitskritischen IT-System durch eine Software (ArcGIS for Server Advance), Hardware (Bull Sequana) und Daten (Naveteq und OSM-Daten) gelöst, die sowohl materielle als auch immaterielle Realgüter darstellen. Die Hardware weist die IT-Sicherheitsmerkmale Härtung und Georedundanz auf. Diese Anforderung wird im Produktivsystem, genannt PU1, umgesetzt. Ein Projektmitarbeiter des Kunden, Polizeihauptkommissar (PHK) Max Mustermann, der beim Polizeipräsidium Haltern arbeitet und dem Projekt angehört, hat Expertenmacht über die Geodatenpflege und kann somit die Anforderungen an die Geodatenpflege beeinflussen. In einem Brainstorming wurde festgestellt, dass diese Anforderung das Risiko birgt, dass der Prozess der Geodatenpflege nicht definiert ist und somit die Anforderung schwer umsetzbar ist.

Obwohl es sich hier um einen fiktiven Fall und fiktive Anforderungen handelt, steht diese Situation stellvertretend für ein komplexes sicherheitskritisches IT-Projekt. Die hellblauen Knoten des Fallgraphs aus jCORA stellen lokale Instanzen dar. Die orangenen Knoten repräsentieren hingegen globale Instanzen, die bereits während der Ontologiekonstruktion mittels Protégé angelegt wurden, wie z. B. Vergabeschritte und Risikomanagementmaßnahmen.

3.3.1.3 Nutzung des CBR-Tools jCORA zur fallbezogenen Ähnlichkeitsberechnung

Wenn eine Fallbeschreibung – gemeint ist weiterhin die Beschreibung eines neuen Projekts – im CBR-Tool jCORA durch den Benutzer abgeschlossen wurde, folgt die *Ähnlichkeitsberechnung* zwischen einem neuen Fall (synonym: Projekt) und den alten Fällen (synonym: Projekten), über die projektmanagementrelevantes Wissen, insbesondere natürlichsprachliches Erfahrungswissen, in der Projektwissensbasis des CBR-Tools vorgehalten wird. Die Ähnlichkeitsberechnung zwischen Fällen wird im CBR-Tool jCORA durchgeführt, indem die Funktion „Aus diesem Fall eine CBR-Anfrage erzeugen“ aufgerufen wird, wie die nachfolgende Abbildung81 illustriert.

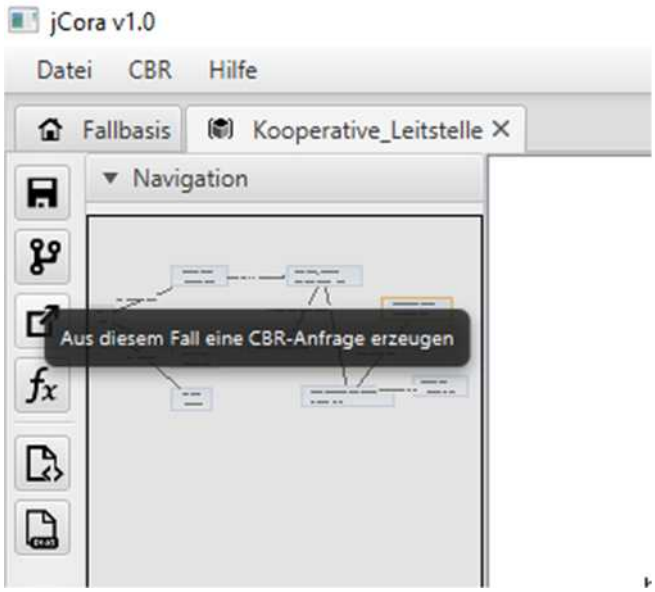


Abbildung 81: CBR-Anfrage in jCORA erzeugen

Die Ähnlichkeitsberechnung erfolgt in einem neuen Fenster, in dem zunächst die Relationen und Attribute als Eigenschaften eines Falls (Projekts) gemäß den eigenen Präferenzen hinsichtlich ihrer Relevanz für die Ähnlichkeitsberechnung anhand von Prozentwerten gewichtet werden können, wie die nachfolgende Abbildung82 illustriert. Darüber hinaus bietet das CBR-Tool jCORA die Möglichkeit, benutzerspezifische Gewichtsprofile zu speichern und für einen späteren Bedarf wieder aufzurufen.

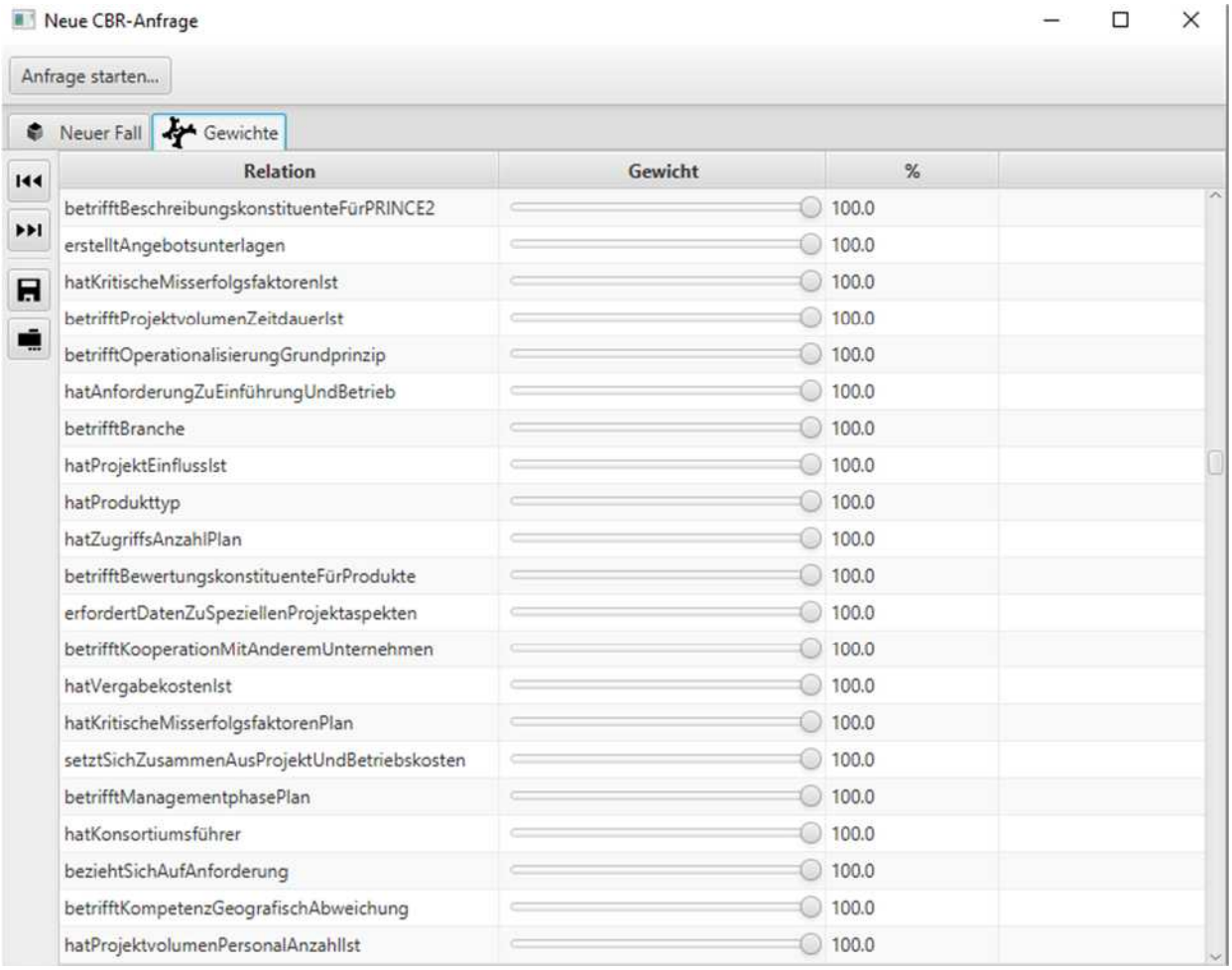


Abbildung 82: Einstellung der Gewichtung der Eigenschaften

Nachdem die Gewichtung festgelegt wurde, kann der Benutzer die Ähnlichkeitsberechnung starten. Dies geschieht, indem die Funktion „Anfrage starten“ aufgerufen wird. Die Ähnlichkeitsberechnung erfolgt für alle Fälle, die in der Fallbasis des CBR-Tools, also seiner Projektwissensbasis, vorliegen. Die nachfolgende Abbildung 83 illustriert die Ergebnispräsentation nach der Ähnlichkeitsberechnung. Es wird angemerkt, dass die Darstellung des Graphs in der Abbildung83 als unzureichend betrachtet werden kann, weil die Bezeichnungen der Werte entlang der x- und y-Achsen fehlen und somit unklar bleibt, welche spezifischen Ergebnisse dargestellt werden. Aus der Benutzerperspektive wäre ein einfacher Ladebalken willkommen, da die Ergebnisse der Ähnlichkeitsberechnung bereits in einem separaten Fenster dargestellt werden.

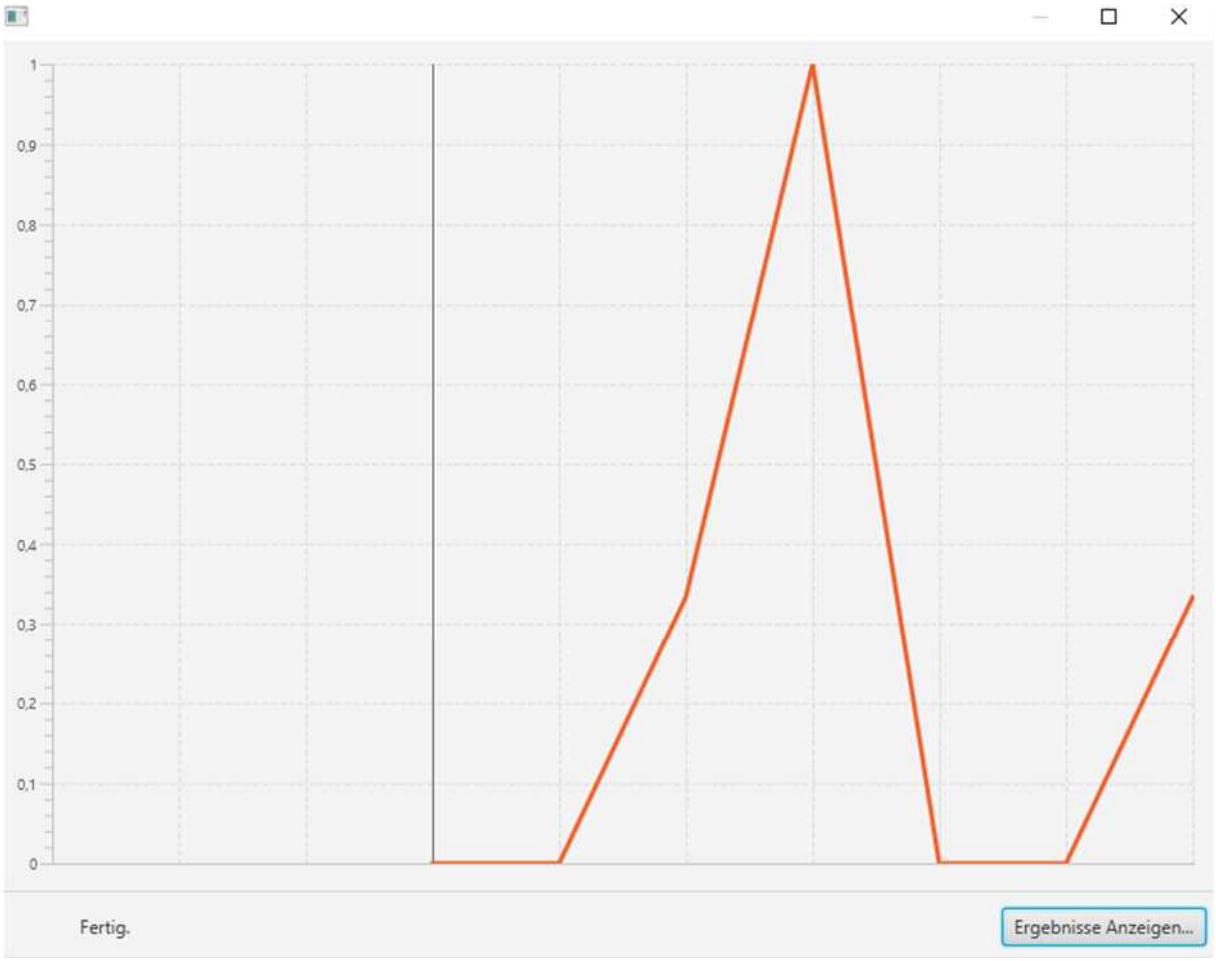


Abbildung 83: Graphische Ergebnispräsentation in jCORA

Nachdem die Ähnlichkeitsberechnung abgeschlossen ist, kann die Funktion „Ergebnisse Anzeigen“ aufgerufen werden, welche die Ergebnisse der Ähnlichkeitsberechnung in dem nachfolgenden Format darstellt:

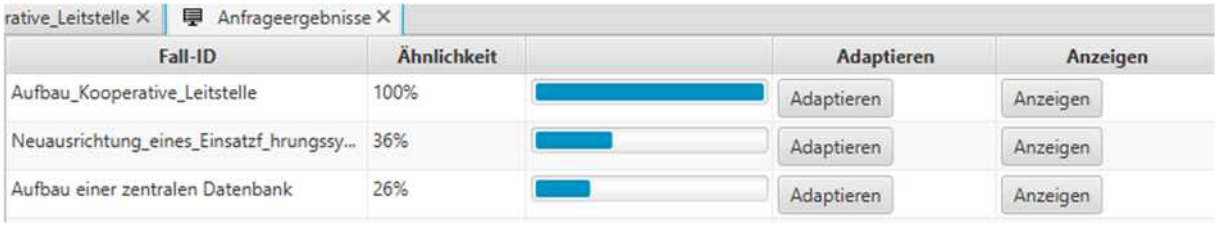


Abbildung 84: Ergebnispräsentation in jCORA

Die berechnete Ähnlichkeit zwischen den Projekten wird prozentual angegeben und von größter Ähnlichkeit zu geringster Ähnlichkeit sortiert. Darüber hinaus wird vom CBR-Tool jCORA eine Filterfunktion angeboten, um nur solche ähnliche Projekte anzuzeigen, die einen bestimmten Ähnlichkeitswert mindestens erreichen. Beispielsweise kann eine Mindestähnlichkeit von 75 % in der Filterfunktion vorgegeben werden.

3.3.1.4 Limitationen des CBR-Tools jCORA

In den bereits vorangegangenen Erläuterungen wurden einige Limitationen (Einschränkungen) des CBR-Tools jCORA für den Einsatz im betrieblichen Umfeld erwähnt. Vgl. auch WEBER et al. (2023), S. 24-25; HERDER/ZELEWSKI/SCHAGEN (2022), S. 44-45; ZELEWSKI/SCHAGEN (2022), S. 51-54; BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 537-541. Dort werden weitere grundsätzliche Limitationen beschrieben, denen das CBR-Tool jCORA hinsichtlich seiner Verwendung im betrieblichen Umfeld unterliegt.

Nachfolgend werden zunächst die wichtigsten Limitationen des CBR-Tools jCORA überblicksartig kurz angesprochen, die auch in anderen, vor allem den vorgenannten Publikationen bereits diskutiert wurden.

Ähnlichkeitsalgorithmus: Der Ähnlichkeitsalgorithmus von jCORA führt zu fehlerhaften Ähnlichkeitsberechnungen, falls globale Instanzen vorschriftswidrig verwendet werden. Eine solche vorschriftswidrige Verwendung liegt vor, wenn globale Instanzen – entgegen ihrer Definition als „global gültig" nicht für alle Projekte unverändert („konstant") verwendet werden, sondern innerhalb der Spezifizierung einzelner Projekte abgewandelt werden, z. B. hinsichtlich der Instanzenwerte. Insofern liegt zwar kein Fehler des Ähnlichkeitsalgorithmus vor, sondern eine zulässige Algorithmusanwendung. Aber Benutzer von jCORA empfinden die dann resultierenden, unplausibel anmutenden Berechnungsergebnisse als einen algorithmischen Fehler. Das Hinzufügen weiterer spezifischer Ähnlichkeitsfunktionen ist nur durch Implementierungen im jCORA-Quellcode möglich. Diese Aufgabe lässt sich nur von IT-Spezialisten, die mit jCORA bestens vertraut sind, fehlerfrei bewältigen. Dies erweist sich als eine erhebliche Limitation hinsichtlich des Einsatzes von jCORA in der betrieblichen Praxis, der oftmals flexible Anpassungen von spezifischen Ähnlichkeitsfunktionen an die Anforderungen in ihrem Projektmanagement-Umfeld verlangt. Des Weiteren erweist sich als unbefriedigend, dass zurzeit noch kein allgemein anerkannter Algorithmus zur Berechnung der Ähnlichkeit zwischen Projekten auf Basis von Projektontologien existiert.

Um die zuvor angesprochene „Fehleranfälligkeit" des Ähnlichkeitsalgorithmus von jCORA bei der Verwendung von *globalen Instanzen* zu verdeutlichen, erfolgt ein kurzer Exkurs. In der sicherheitskritischen IT-Projekt-Ontologie wurden einige globale Instanzen konstruiert (z. B. die Vergabeschritte und auch die Projektmanagementmethode PRINCE2), die jedoch bei der Anwendung in jCORA zu falschen Ähnlichkeitswerten führen, was auch bereits von WEBER et al. (2021), S. 26, angemerkt wurde. Aufgrund dieses Umstands wurde in den folgenden drei Fällen nur im Fall „Neuausrichtung eines Einsatzführungssystems" eine globale Instanz verwendet, um nicht zu falschen Werten bei der Ähnlichkeitsberechnung zu führen.

Globale Instanzen können zwar in einem Ontologie-Editor wie Protegé erstellt werden, jedoch können sie in jCORA im Zusammenhang mit der Ähnlichkeitsberechnung zu Fehlern führen, wenn sie in jCORA während der Spezifizierung von Projektinstanzen („Fällen") durch das Hinzufügen neuer Attribute oder Relationen „lokal" (also instanzenbezogen) modifiziert werden.

Dies könnte zu einem Widerspruch zwischen der globalen Instanzdefinition in Protégé einerseits und der lokalen Instanzmodifikation in jCORA andererseits führen. Unter dieser Prämisse wären globale Instanzen nicht per se fehleranfällig, sofern gewährleistet wird, dass sie in jCORA niemals auf die zuvor erwähnte Weise „modifiziert“ werden. Jedoch ist es ebenso legitim, Kritik zu äußern, dass durch den Verzicht auf die Modifikation von globalen Instanzen potenziell ein Verlust der verfügbaren oder konstruierten sprachlichen Ausdrucksmittel einhergeht.

Die nachfolgende Abbildung85 veranschaulicht das Problem, wenn ausgehend vom Fall „Neuausrichtung eines Einsatzführungssystems“ die Ähnlichkeit zu anderen Fällen und zu sich selbst berechnet wird. jCORA gibt die Ähnlichkeit des Falles „Neuausrichtung eines Einsatzführungssystems“ zu sich selbst mit 98 % an, was nicht korrekt ist.

Abbildung 85: Fehlerhaftes Ergebnis der Ähnlichkeitsberechnung bei der Verwendung von globalen Instanzen

Abschließend muss auch der in jCORA berechnete Ähnlichkeitswert kritisch bewertet werden. In diesem Beitrag wird in Kapitel 3.3.3.2 die Ähnlichkeit zwischen den drei Fällen berechnet. Die Berechnung basiert auf der in jCORA standardmäßig eingesetzten, sogenannten universellen Ähnlichkeitsfunktion. Um diese Einschränkung zu überwinden, wird später in diesem Beitrag eine spezifische Ähnlichkeitsfunktion implementiert, die mit Hilfe der Word2Vec-Technik die Ähnlichkeit zwischen Wörtern berechnen kann. Die Implementierung als Serverless-Funktion ermöglicht das Hinzufügen weiterer spezifischer Ähnlichkeitsfunktionen, ohne dass der vorhandene Quellcode einer bestehenden Serverless-Funktion geändert werden muss.

Ähnlichkeitstabellen können zwar geladen werden, aber eine spezifische Ähnlichkeitsfunktion muss im Quellcode implementiert werden, was in einer monolithischen Anwendung nicht trivial ist, da es sich um einen Quellcodeblock handelt. Zudem ist der Quellcode nicht durch ein technisches Handbuch dokumentiert. Obwohl der Quellcode gut dokumentiert ist, wäre ein Architekturplan des CBR-Tools jCORA wünschenswert, um die Beziehungen zwischen den Klassen dieses CBR-Tools besser zu verstehen.

Des Weiteren ist die universelle Ähnlichkeitsfunktion $sim_{numeric}$ hinsichtlich ihrer Plausibilität kritisch zu hinterfragen, da die Werte in der Ontologie je nach Kontext (beispielsweise *TCVBC* oder *hatAlter*) stark variieren und die Ähnlichkeit auf der Basis des größten und des kleinsten Wertes berechnet wird. Auch mit den genannten Einschränkungen in der Fallspezifizierung, die als zusätzliche Einschränkungen von jCORA verstanden werden können, zeigen die Ergebnisse in Kapitel 3.3.3.2, dass eine tendenzielle Ähnlichkeit zwischen den Fällen berechnet werden kann, um nützliche Erkenntnisse für die Wiederverwendung von Erfahrungswissen aus sicherheitskritischen IT-Projekten zu gewinnen. Die nachfolgenden Erläuterungen

der Fälle umfassen nicht alle Bereiche eines Falles, da die Erläuterungen sonst einen unverhältnismäßig großen Umfang einnehmen würden. Vielmehr werden ausgewählte Bereiche der Fälle erläutert, wie z. B. die Anforderungen der Leistungsbeschreibungen, die als Grundlage für die verwendeten Realgüter dienen.

Ausdrucksmächtigkeit der zugrunde gelegten Ontologie: Die Leistungsfähigkeit von jCORA hängt wesentlich von der zugrunde liegenden Ontologie ab, insbesondere von deren Ausdrucksstärke. Einschränkungen in der Ontologie, beispielsweise aufgrund von Beschränkungen des Ontologieeditors wie Protégé, können sich negativ auf die Leistungsfähigkeit von jCORA auswirken. Dazu gehört beispielsweise, dass – wie bereits erwähnt – nur zweistellige nicht-taxonomische Relationen in Protégé und darauf aufbauend auch in jCORA verwendet werden können, obwohl die betriebliche Praxis zuweilen auch zumindest dreistellige (oder noch „höherstellige“) nicht-taxonomische Relationen als sprachliche Ausdrucksmittel für eine „natürliche“ Modellierung der jeweils relevanten betriebswirtschaftlichen Sachverhalte erfordert. Dies betrifft z. B. nicht-taxonomische Relationen für Kompetenzrelationen, die als Relationsstellen zumindest einen Kompetenzträger (Akteur), eine Kompetenzart und das Ausmaß der Kompetenzerfüllung umfassen sollten. Hinzukommen könnten beispielsweise Angaben zum Nachweis des Verfügens über eine Kompetenz (Zertifikate) und Angaben zum Zeitraum, während dessen die Kompetenz erworben – oder auch infolge Nichtgebrauchs eventuell verlernt wurde.

Visualisierung von instanziierten Projekt-Ontologien mittels Fallgraphen: Die Einschränkungen, die hinsichtlich der Visualisierung festzustellen sind, spiegeln sich in der grundsätzlich mangelhaften Usability des CBR-Tools jCORA wider. Die Einschränkungen der Usability für die Verwendung dieser Software im betrieblichen Umfeld werden insbesondere von WEBER et al. (2023), S. 53-56, näher beschrieben.

Hilfetools zu jCORA: Zwar existieren inzwischen einige Publikationen zu jCORA. Jedoch stellen diese Publikationen keinen adäquaten Ersatz für eine benutzerbezogene Online-Hilfe dar, die eine Verwendung von jCORA im betrieblichen Umfeld erheblich unterstützen würde. Erwähnenswert ist in dieser Hinsicht die Publikation SCHAGEN et al. (2022), in der die Entwicklung eines E-Learning-Moduls für das benutzerfreundliche Einarbeiten in die Anwendung von jCORA vorgestellt wird.

Adaption: Die Anpassung (Adaption) der Lösungen für ähnlichste alte Projekte (Fälle) an die Beschreibungen neuer Projekte (Fälle) stellt nicht nur für das CBR-Tool jCORA im Speziellen, sondern auch für CBR-Systeme im Allgemeinen ein derzeit noch nicht einmal ansatzweise zufriedenstellend gelöstes Problem für den Erfolg versprechenden Einsatz von Case-based Reasoning in der betrieblichen Praxis hinsichtlich der Wiederverwendung von Erfahrungswissen im Projektmanagement dar. Darauf wird in Kürze näher eingegangen, und zwar mit Bezug auf die speziellen Limitationen des CBR-Tools jCORA.

User Interface: Die Limitationen bezüglich der Usability von jCORA wird von WEBER et al. (2023) ausführlicher diskutiert. Dort wird über Usability-Tests berichtet, die unter anderem mit Projektleitern aus dem Bereich von sicherheitskritischen IT-Projekten durchgeführt wurden und jCORA kein insgesamt überzeugendes Zeugnis ausstellten; vgl. WEBER et al. (2023), S. 39-56.

Zur Verdeutlichung wird im Folgenden nur auf die Limitation der Adaption in exemplarischer Weise vertiefend eingegangen, weil die Adaption der Lösungen für ähnlichste alte Projekte an die Beschreibungen neuer Projekte für das Case-based Reasoning eine sehr wichtige Rolle spielt. Im Sinne der Reuse-Phase des früher vorgestellten CBR-Zyklus können ein altes Projekt und seine Wissenskomponenten (Projektlösung und Projektbewertung) wiederverwendet werden, indem die Funktion „Adaptieren“ aufgerufen wird. Eine Adaptionsregel wird im Fenster „Vorhandene Regeln“ ausgewählt und durch die Funktion „Anwenden“ ausgeführt. Dadurch wird die ausgewählte Anpassungsregel auf die Lösung für ein neues Projekt angewendet.

Die nachfolgende Abbildung86 illustriert, dass das CBR-Tool jCORA gegenwärtig nur die eine Anpassungsregel „Kopiere Lösung“ besitzt. Mittels dieser Adaptionsregel wird die gesamte Lösung eines alten Projektes in die Lösung für ein neues Projekt kopiert. Diese Kopierfunktion übernimmt die Projektlösung für das ähnlichste alte Projekt als Lösungsvorschlag für ein neues Projekt, ohne Anpassungen für das neue Projekt vorzunehmen. Die notwendige Adaption der Projektlösung für ein ähnlichstes altes Projekt an das betrachtete neue Projekt muss daher „manuell“ erfolgen. Diese Adaptionsregel wird auch als „Nulladaption“ bezeichnet; vgl. WILKE/BERGMANN (1998), S. 500.

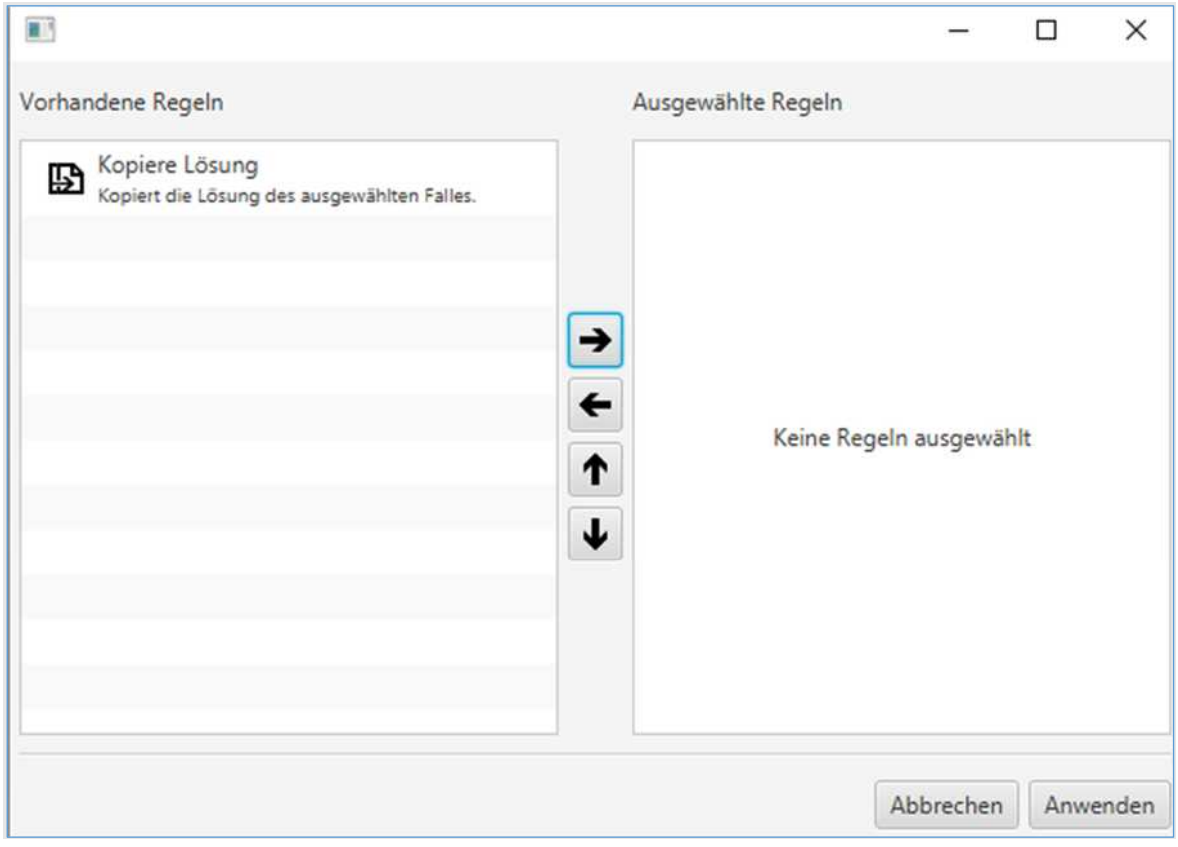


Abbildung 86: Adaptionsfenster in jCORA

Die Nulladaption ist stark umstritten, weil die Kopie der Lösung für ein altes Projekt als Lösungsvorschlag für ein neues Projekt der oftmals definitorisch hervorgehobenen „Einzigartigkeit“ eines Projektes widerspricht. Stattdessen wird zuweilen vorgeschlagen, die Lösung für ein altes Projekt als Ausgangslösung zu verwenden, um auf dieser Grundlage mittels „manueller“ Anpassungen zu einer Lösung für das jeweils betrachtete neue Projekt zu gelangen. Diese „manuelle“ Anpassung der Lösungen für sicherheitskritische Projekte widerspricht jedoch der Anforderung, neue sicherheitskritische IT-Projekte möglichst weitgehend mit „intelligenten“, KI-basierten Wissensmanagementsystemen *computerunterstützt* managen zu können.

Allerdings stellt die automatische Anpassung von Lösungen für alte an neue Projekte in einem CBR-Tool ein *Forschungsfeld* dar, das noch umfangreiche und tiefschürfende weitere Untersuchungen erfordert. Dieses Forschungsfeld leidet im State of the Art zu CBR-Tools derzeit noch unter viel zu großen Forschungs- und Implementierungslücken.

Das CBR-Tool jCORA bietet zwar bereits eine generische Anpassungsfunktion „adapt“, die genutzt werden kann, um benutzerspezifische Adaptionsregeln zu definieren; vgl. ZELEWSKI/SCHAGEN (2022), S. 40-41; BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 534-536. Aber diese generische Anpassungsfunktion reicht noch nicht aus, weil sie weder benutzer- noch projekt- noch branchenspezifische Adaptionsregeln definiert. In dieser Hinsicht weist das CBR-Tool jCORA – wie auch andere CBR-Tools – eine gravierende Anwendungslücke auf.

In den nachfolgenden Ausführungen wird ausführlicher auf eine besondere Limitation des CBR-Tools jCORA eingegangen, die in der einschlägigen Fachliteratur (mit Ausnahme von SETHUPATHY (2024), S. 341-346) bislang noch nicht behandelt wurde. Es handelt sich um die *monolithische Anwendungsstruktur* von jCORA. Sie stellt für die Praxistauglichkeit von jCORA im betrieblichen Umfeld eine zentrale Schwachstelle dar. Sie wird hier zunächst näher erläutert. Ein umfangreicher, Cloud-basierter Ansatz zur Überwindung dieser Schwachstelle wird in diesem Beitrag an späterer Stelle in Kapitel 4 präsentiert.

Die zentrale Bedeutung der monolithischen Anwendungsstruktur von jCORA beruht darauf, dass durch aktuelle technologische Entwicklungen im Cloud-Bereich monolithische Anwendungen im betrieblichen Umfeld zunehmend abgelöst werden; vgl. FRANK/SCHUMACHER/TAMM (2019), S. 6; FRITZSCH et al. (2019), S. 128-129. Die Nachteile einer monolithischen Anwendung in einem betrieblichen Umfeld werden u. a. von OLIVEIRA ROCHA (2021), S. 4-9, ausführlich beschrieben. Sie unterstützen die Annahme, dass die aktuellen technologischen Entwicklungen im Cloud-Bereich monolithische Anwendungen im betrieblichen Umfeld weitgehend ablösen werden.

Obwohl die Limitation des CBR-Tools jCORA hinsichtlich seiner monolithischen Anwendungsstruktur eine gravierende Behinderung für seine Verwendung im betrieblichen Umfeld darstellt, wurde sie bisher in keiner wissenschaftlichen Publikation behandelt (mit Ausnahme von SETHUPATHY (2024): siehe oben). Von HERDER/ZELEWSKI/SCHAGEN (2022), S. 44-45, wird zwar die Kritik geäußert, dass von jCORA keine „praxistauglichen“ Schnittstellen ange-

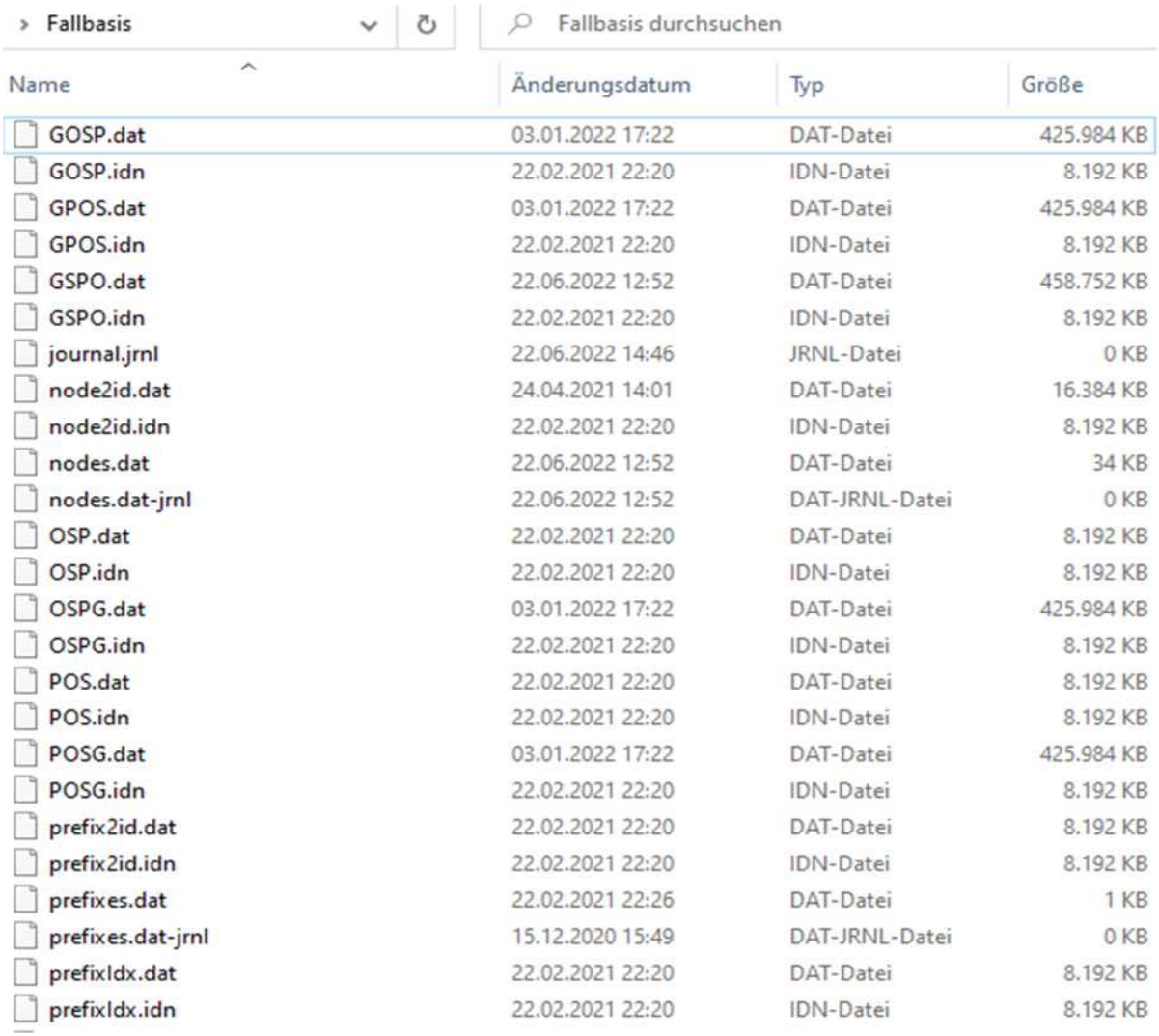
boten werden. Jedoch wird die zugrunde liegende Limitation durch die monolithische Anwendungsstruktur von jCORA nicht angesprochen. Dort wird noch nicht berücksichtigt, dass mithilfe der Auflösung einer monolithischen Anwendungsstruktur neuartige Schnittstellen angeboten werden können, die sich von weiteren – vor allem in der betrieblichen Praxis weit verbreiteten – Anwendungssoftwares, wie z. B. von Microsoft-Office-Produkten, nutzen lassen. Dies wird im Folgenden näher erläutert. Jedoch sei vorweggenommen, dass für jede neuartige Schnittstelle eine eigene Serverless-Funktion implementiert werden muss. Dies bedeutet einen nicht zu unterschätzenden Ressourceneinsatz für Cloud-basierte Reimplementierung des CBR-Tools jCORA. Sie verheißt jedoch auch einen hohen Zusatznutzen im Hinblick auf größere Anwendungsflexibilität und neuartigen Schnittstellen zu betrieblich weithin etablierten Anwendungssoftwares.

Das CBR-Tool jCORA besitzt eine monolithische Anwendungsstruktur, in der die Funktionalitäten in einer einzigen Applikation gebündelt werden, und ist in der 1-Tier-Architektur einzuordnen, sodass keine Datenhaltungsebene existiert. Beispielsweise hätte eine Datenhaltungsebene mittels einer gängigen Datenbank implementiert werden können, um die Präsentationsebene und die Datenhaltungsebene zu trennen (2-Tier-Architektur). Protégé unterstützt prinzipiell die Nutzung einer Datenbank als Backend-Komponente, sodass eine beliebige Anwendung auf die Datenbank zugreifen könnte und die in Protégé konstruierten Ontologien aufgerufen werden können. Die Unterstützung von Datenbanken in Protégé sowie die notwendigen Anbindungsformalien zu einer Datenbank (Connection-Strings) werden in PROTÉGÉ (2010) beschrieben. Es wäre in jCORA denkbar gewesen, die zugrunde liegende Ontologie sowie die Projektwissensbasis in einer gängigen Datenbank abzulegen und mittels Schnittstellen in jCORA einzubinden. Mittels Datenbanktriggern hätten die notwendigen Formalien – wie z. B. das Vorliegen der Fallstruktur (Projektstruktur) mit Fallbeschreibung, Falllösung und Fallbewertung – automatisch geprüft werden können. Dadurch hätten neben der Java-Technologie – die für das CBR-Tool jCORA genutzt wird – auch Datenbanktechnologien wie SQL genutzt werden können, um beispielsweise automatisierte Prüfszenarien in der Datenbank (Projektwissensbasis) zu konstruieren. Jedoch wäre bei Vorliegen dieser Trennung weiterhin die Einschränkung einer monolithischen Anwendungsstruktur gegeben. Da diese grundlegende Trennung in jCORA fehlt, stellt die monolithische Anwendungsstruktur von jCORA eine fundamentale Limitation hinsichtlich Nutzung dieses CBR-Tools im betrieblichen Umfeld dar.

Die grundlegende Trennung zwischen Präsentationsebene und Datenhaltungsebene würde die Wartbarkeit des CBR-Tools jCORA unterstützen, da zwischen den Schichten die Kopplung möglichst geringgehalten wird und eindeutige Schnittstellen zwischen den beiden Schichten existieren; vgl. RAU (2016), S. 304. Diese Unterteilung würde auch die Portierbarkeit unterstützen, sodass es einfacher möglich wäre, die Datenbank auszutauschen oder eine andere Präsentationsebene bei gleichbleibender Datenhaltungsebene zu verwenden.

Auch wenn jCORA diese fundamentale Trennung zwischen Datenhaltungsebene und Präsentationsebene unterstützen würde, wäre das CBR-Tool aus Implementierungssicht jedoch weiterhin eine monolithische Anwendung. In jCORA werden die Fälle der Fallbasis (Projekte der

Projektwissensbasis) in einer proprietären .dat- und .idn-Datei gespeichert. Die nachfolgende Abbildung87 illustriert die .dat- und .idn-Dateien, die bei jCORA „automatisch“ in dem Ordner „Fallbasis“ angelegt werden.



Name	Änderungsdatum	Typ	Größe
GOSP.dat	03.01.2022 17:22	DAT-Datei	425.984 KB
GOSP.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
GPOS.dat	03.01.2022 17:22	DAT-Datei	425.984 KB
GPOS.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
GSPO.dat	22.06.2022 12:52	DAT-Datei	458.752 KB
GSPO.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
journal.jrnl	22.06.2022 14:46	JRNL-Datei	0 KB
node2id.dat	24.04.2021 14:01	DAT-Datei	16.384 KB
node2id.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
nodes.dat	22.06.2022 12:52	DAT-Datei	34 KB
nodes.dat-jrnl	22.06.2022 12:52	DAT-JRNL-Datei	0 KB
OSP.dat	22.02.2021 22:20	DAT-Datei	8.192 KB
OSP.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
OSPG.dat	03.01.2022 17:22	DAT-Datei	425.984 KB
OSPG.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
POS.dat	22.02.2021 22:20	DAT-Datei	8.192 KB
POS.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
POSG.dat	03.01.2022 17:22	DAT-Datei	425.984 KB
POSG.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
prefix2id.dat	22.02.2021 22:20	DAT-Datei	8.192 KB
prefix2id.idn	22.02.2021 22:20	IDN-Datei	8.192 KB
prefixes.dat	22.02.2021 22:26	DAT-Datei	1 KB
prefixes.dat-jrnl	15.12.2020 15:49	DAT-JRNL-Datei	0 KB
prefixIdx.dat	22.02.2021 22:20	DAT-Datei	8.192 KB
prefixIdx.idn	22.02.2021 22:20	IDN-Datei	8.192 KB

Abbildung 87: .dat- und .idn-Dateien von jCORA in dem Ordner *Fallbasis*

Eine .dat-Datei ist eine generische Datendatei, die von einer bestimmten Software erstellt wurde. Meistens bietet die zugehörige Software die einzige Möglichkeit, diese Datendatei zu öffnen, was wiederum die monolithische Anwendungsstruktur verschärft. Die Dateiendung .idn ist nicht allgemein festgelegt. Aufgrund fehlender technischer Dokumentationen des CBR-Tools jCORA ist nicht klar, wofür die .idn-Dateien erstellt werden. Es wurde jedoch beobachtet, dass bei der Konstruktion von Fällen automatisch .idn- und .dat-Dateien im Ordner „Fallbasis“ erstellt werden.

Neben der grundsätzlichen Einschränkung durch die 1-Tier-Architektur besitzen monolithische CBR-Tools („Anwendungen“) wie jCORA weitere generelle Nachteile hinsichtlich der Nutzung im betrieblichen Umfeld, die in den nachfolgenden Punkten kurz zusammengefasst werden:

- Technologie: Bei der Anpassung einer monolithischen Anwendungsstruktur muss die gesamte betroffene Software auf Funktionsfähigkeit getestet werden. Dies bedeutet, dass die Entwicklung einer Software komplexer und langsamer wird; vgl. FRITZSCH et al. (2019), S. 129; RAU (2016), S. 304. Des Weiteren haben Änderungen in einer monolithischen Anwendungsstruktur Auswirkungen auf die gesamte Anwendung, was zu zeitaufwändigen Anpassungen führt und dadurch die Kosten erhöht; vgl. FRANK/SCHUMACHER/TAMM (2019), S. 161-162. Die hohe Komplexität sowie die vielen Abhängigkeiten innerhalb der monolithischen Anwendungsstruktur erschweren die Lokalisierung eines möglichen Fehlers in einer Software; vgl. OLIVEIRA ROCHA (2021), S. 6 u. 13. Testverfahren müssen komplexer angelegt werden, um jegliche Fehlerfälle, die durch eine Änderung der Software entstehen können, aufgrund der internen Abhängigkeiten auszuschließen. Eine monolithische Anwendungsstruktur ist in der Regel ausschließlich in einer Programmiersprache verfasst. Technologische Fortschritte, die beispielsweise durch neuartige Programmiersprachen entstehen, können nicht berücksichtigt werden; vgl. OLIVEIRA ROCHA (2021), S. 15; FRITZSCH et al. (2019), S. 129.
- Bereitstellung: Jegliche Änderungen einer Software mit einer monolithischen Anwendungsstruktur erfordern eine komplette Bereitstellung (Deployment) der Software. Bei jCORA müsste die Software an allen betroffenen Arbeitsplätzen mittels einer Softwareverteilung oder durch eine Vor-Ort-Installation bereitgestellt werden. Softwareverteilungstools (Application Release Automation) sind Automatisierungstools zur Verteilung von Software auf unterschiedliche Zielsysteme; vgl. PFITZINGER/JESTÄDT (2017), S. 582. Beide Varianten würden Kosten für die Bereitstellung und Ausfallzeiten während der Bereitstellung verursachen. Des Weiteren kann eine Anpassung an den betroffenen Arbeitsplätzen notwendig sein. Beim Java-basierten CBR-Tool jCORA kann es beispielsweise notwendig sein, die Java Runtime oder die Umgebungsvariable zu aktualisieren, was wiederum zusätzlichen Aufwand erfordert und eine weitere Fehlerquelle darstellt. Eine Umgebungsvariable (Path Parameter) bezeichnet konfigurierbare Pfade in einem Betriebssystem zu bestimmten Softwares. Der Nachteil bei der Definition einer Umgebungsvariable ist, dass die Umgebungsvariable jeweils über die Systemsteuerung eines Clients angepasst werden muss. Dies würde bei einer verteilten Client-Umgebung über mehrere Standorte einen erhöhten Aufwand bedeuten, da eine Anpassung in jeder Client-Umgebung erfolgen müsste.
- Mobilität: Zurzeit ist das CBR-Tool jCORA nicht auf mobilen Endgeräten lauffähig. Dies entspricht dem gängigen Muster bei monolithischen Anwendungen, da sie in der Regel für Desktop-Clients entwickelt werden. Daher wäre die Entwicklung einer eigen-

ständigen Softwarevariante erforderlich, um ihre mobile Ausführbarkeit zu ermöglichen. Die Nutzung einer Software auf mobilen Endgeräten stellt eine wichtige Anforderung hinsichtlich ihres Einsatzes in der betrieblichen Praxis dar. Dies wird dadurch begründet, dass im Jahr 2021 der Anteil der mobilen Internetnutzer in Deutschland bei 82 % lag; vgl. INITIATIVE D21 (2022), S. 14-15. Der Anteil der Besitzer mobiler Geräte in der Bevölkerung in Deutschland betrug im Jahr 2021 rund 88,8 %; vgl. TENZER (2022). Der Wert bezieht sich auf Personen ab 14 Jahren, die ein Smartphone oder Handy im Haushalt besitzen. Das CBR-Tool jCORA, das eine Java-basierte Software darstellt, lässt sich auf mobilen Geräten mit dem Betriebssystem (iOS) standardmäßig nicht nutzen. Denn das Betriebssystem iOS unterstützt die Programmiersprache Java nicht, sondern nur seine eigene Programmiersprache Swift. Bei mobilen Android-Geräten wäre eine Nutzbarkeit denkbar, da Android die Programmiersprache Java als Standardprogrammiersprache unterstützt; vgl. MAWLOOD-YUNIS (2022), S. 55. Jedoch existieren Technologien, um beide mobile Endgerättypen, die mehr als 99 % des Marktanteils repräsentieren (KANTARWORLDPANEL (2022), gleichermaßen zu unterstützen.

- Skalierbarkeit: Einzelne Komponenten des CBR-Tools jCORA können nicht skaliert werden. Eine Skalierung erfolgt in der Regel durch die Duplizierung der gesamten Software beispielweise auf einen weiteren – unter Umständen auch nur virtuellen – Computer; vgl. OLIVEIRA ROCHA (2021), S. 15; FRITZSCH et al. (2019), S. 129. Dieser Skalierungsansatz stellt jedoch einen ineffizienten Ansatz dar, um auf schnell ändernde Workloads zu reagieren und gleichzeitig hinsichtlich der Ressourcennutzung optimal zu bleiben; vgl. FRITZSCH et al. (2019), S. 129. Als ein Kriterium zur Bewertung von Ineffizienz kommt das Verhältnis zwischen eingesetzten Hardwareressourcen und Berechnungszeiten in Betracht. Ein Optimalitätskriterium kann in diesem Kontext die Erzielung maximaler Leistung bei möglichst unveränderter Latenz sein. Dieses Kriterium zielt darauf ab, die höchste Leistung zu erreichen, ohne die Latenz zu beeinträchtigen. Dies bedeutet, dass eine große Anzahl von Anfragen schnell bearbeitet werden kann, ohne die Antwortzeit (Latenz) unnötig zu verlangsamen.

Softwares mit monolithischer Anwendungsstruktur können zwar zumindest in frühen Phasen der Softwareentwicklung den Vorteil besitzen, die Software schnell zu entwickeln und den kognitiven Aufwand für das Codemanagement und das Deployment zu reduzieren; Vgl. HARRIS (2022); OLIVEIRA ROCHA (2021), S. 10-11. Insbesondere im Bereich des „Rapid Prototyping“ könnte eine Software mit monolithischer Anwendungsstruktur zumindest helfen, einen lauffähigen Prototyp rasch zu entwickeln. Durch die Verwendung ausschließlich einer Codebasis (hier die Programmiersprache Java) kann die gesamte Entwicklung in einer Entwicklungsumgebung, beispielsweise Eclipse, erfolgen. Dies könnte die Entwicklung einer Software vereinfachen, insbesondere zu Beginn des Entwicklungsprozesses; vgl. HARRIS (2022). Weitere Vorteile von monolithischen Anwendungsstrukturen werden in HARRIS (2022) und OLIVEIRA ROCHA (2021), S. 9-13, angeführt. Insgesamt kommen die aktuellen Publikationen jedoch zum Schluss, dass die Nachteile von Software mit monolithischen Anwendungsstrukturen für den

Einsatz in einem betrieblichen Umfeld überwiegen. Siehe hierzu beispielsweise OLIVEIRA ROCHA (2021), S. 13; FRANK/SCHUMACHER/TAMM (2019), S. 153-154.

Bei unternehmensrelevanten Softwares stellen monolithische Anwendungsstrukturen insbesondere auch einen organisatorischen Engpass dar; vgl. FRANK/SCHUMACHER/TAMM (2019), S. 153-154. Ihrer Ansicht nach bestimmt die Leistungsfähigkeit einer Software wesentlich die Leistungsfähigkeit eines Geschäftsmodells, das auf dieser Software beruht. Für ein ontologiegestütztes CBR-System, wie das hier betrachtete CBR-Tool jCORA, das für die Wiederverwendung von Erfahrungswissen im betrieblichen Umfeld des Projektmanagements genutzt wird, stellt eine monolithische Anwendungsstruktur aufgrund der vorgenannten Schwierigkeiten (Limitationen) ein erhebliches betriebswirtschaftliches Risiko dar.

Im Idealfall funktioniert eine Software mit monolithischer Anwendungsstruktur zwar fehlerfrei. Kommt es jedoch zu einem Fehler, beispielsweise in einer der vier Phasen des CBR-Zyklus, oder wird eine Teilkomponente der monolithischen Anwendungsstruktur, beispielsweise die Fallbasis (Projektwissensbasis) des CBR-Tools jCORA, überlastet, ist es sehr wahrscheinlich, dass die gesamte Software einen Fehlerfall produziert und somit nicht mehr einsetzbar ist. Die Verfügbarkeit der gesamten Software ist damit gefährdet. Ihre Nutzung als zentrales Wissensmanagement-System könnte bei einem „Gesamtausfall“ zu einem erheblichen betrieblichen Risiko führen.

Die monolithische Anwendungsstruktur des CBR-Tools jCORA erschwert die flexible Weiterentwicklung dieses CBR-Tools und eine schnelle Bereitstellung von zusätzlichen Funktionalitäten erheblich, eventuell wird eine solche Weiterentwicklung wegen eines „prohibitiv“ hohen Aufwands sogar grundsätzlich vereitelt. Daher wäre eine nicht-monolithische Anwendungsstruktur, bei der einzelne Funktionen (beispielsweise für einzelne Phasen des CBR-Zyklus, wie etwa die derzeit noch nicht zufriedenstellend abgedeckte Reuse-Phase) unabhängig von anderen Komponenten (wie beispielsweise dem User Interface) entwickelt und bereitgestellt werden könnten, wesentlich flexibler. Eine mögliche Lösung für das zuvor skizzierte Problem stellen sogenannte Serverless-Funktionen dar, die im späteren Verlauf dieses Beitrags ausführlich erläutert werden.

Neben den genannten Nachteilen von Softwares mit monolithischen Anwendungsstrukturen entsprechen solche „monolithischen“ Softwares nicht mehr dem aktuellen Stand der „modernen“ Softwaretechnologie, insbesondere in Bezug auf ökonomische Aspekte; vgl. FRANK/SCHUMACHER/TAMM (2019), S. 167. Die Infrastrukturautomatisierung für den Einsatz von Softwares im betrieblichen Umfeld hat sich in den letzten Jahren stark entwickelt; vgl. FOWLER/LEWIS (2015), S. 19. Die technologischen Entwicklungen im Cloud-Computing – insbesondere die Angebote von „Hyperscalern“ wie Amazon Web Services (AWS) und Microsoft Azure – haben die operative Komplexität von Entwicklung, Bereitstellung und Betrieb von Softwares mit nicht-monolithischen Anwendungsstrukturen (Microservices) erheblich reduziert und den Einsatz von Softwares mit monolithischen Anwendungsstrukturen im betrieblichen Umfeld hinsichtlich des „State of the Art“ überholt; vgl. JAMSHIDI et al. (2018), S. 26-27; FOWLER/

LEWIS (2015), S. 5. In diesem Zusammenhang entwickeln sich in der betrieblichen Praxis eingesetzte Softwares zunehmend zu Cloud-native-Anwendungen, die ausschließlich in der Cloud bereitgestellt und weiterentwickelt werden. Die Entwicklung eines Cloud-basierten ontologiegestützten Case-based-Reasonings als ein spezieller Anwendungsfall von solchen Cloud-native-Anwendungen wird in Kapitel 4 ausführlich erläutert.

3.3.2 Beschreibung der Fälle zur Repräsentation von drei Praxisbeispielen

3.3.2.1 Vorbemerkungen zu den drei Praxisbeispielen

Anschließend werden drei Praxisbeispiele mithilfe des CBR-Tools jCORA konstruiert und später hinsichtlich ihrer Ähnlichkeiten analysiert. Diese Praxisbeispiele sollen einerseits verdeutlichen, wie sich Projekte als „Fälle“ in einem ontologiegestützten CBR-System repräsentieren und lassen und wie das Erfahrungswissen aus alten, bereits durchgeführten Projekten für die Planung, Durchführung und Kontrolle neuer Projekte mittels eines CBR-Tools wie jCORA wiederverwendet werden kann. Andererseits wird anhand dieser drei Praxisbeispiele veranschaulicht, wie sich sicherheitskritische IT-Projekte innerhalb eines ontologiegestützten CBR-Systems wie dem CBR-Tool jCORA konkret erfassen lassen.

Die nachfolgenden drei Fälle – synonym: Praxisbeispiele und sicherheitskritische IT-Projekte – stellen zwar keine real existierenden, sicherheitskritischen IT-Projekte dar. Sie unterstützen es jedoch, die Ausgestaltung eines sicherheitskritischen IT-Projekts zu erkennen, das ein vorangegangenes Vergabeverfahren besitzt und Anforderungen an ein sicherheitskritisches IT-System erfüllt, die in die zuvor genannten Anforderungsbereiche einzuordnen sind. Die Fallbezeichnungen sind bewusst an gängige Projekte angelehnt. Die nachfolgende Tabelle 54 führt die in diesem Beitrag untersuchten drei Fälle an.

Fall Nr.	**Fall-Bezeichnung**
1	Neuausrichtung eines Einsatzführungssystems der Polizei
2	Aufbau einer kooperativen Leitstelle
3	Aufbau einer zentralen Datenbank für Ermittlungen

Tabelle 54: Praxisbeispiele für sicherheitskritische IT-Projekte

Neben den in Kapitel 3.3.1.4 genannten *grundsätzlichen* Limitationen des CBR-Tools jCORA sind weitere Einschränkungen zu nennen, die hinsichtlich der nachfolgend betrachteten drei Praxisbeispiele („Fälle“) für sicherheitskritische IT-Projekte in *fallspezifischer* Hinsicht aufgefallen sind.

Die drei betrachteten Fälle stellen streng genommen keine vollständigen Fälle dar, sondern umfassen lediglich einige Anforderungen aus sicherheitskritischen IT-Projekten. Diese Einschränkung musste erfolgen, weil das CBR-Tool jCORA bei einer umfassenderen Fallkonstruktion

und anschließender Ähnlichkeitsberechnung zuweilen in einen Fehler läuft, wenn zu viele oder zu umfangreiche Fälle (Instanzen) berücksichtigt werden; vgl. die nachfolgende Abbildung88. Daher musste hinsichtlich der Anzahl und des Umfangs der konstruierten Instanzen ein Kompromiss gefunden werden, um einerseits aussagekräftige Fälle zu konstruieren, andererseits aber auch nicht zu viele und nicht zu umfangreiche Instanzen anzulegen, die dann unbrauchbar wären.

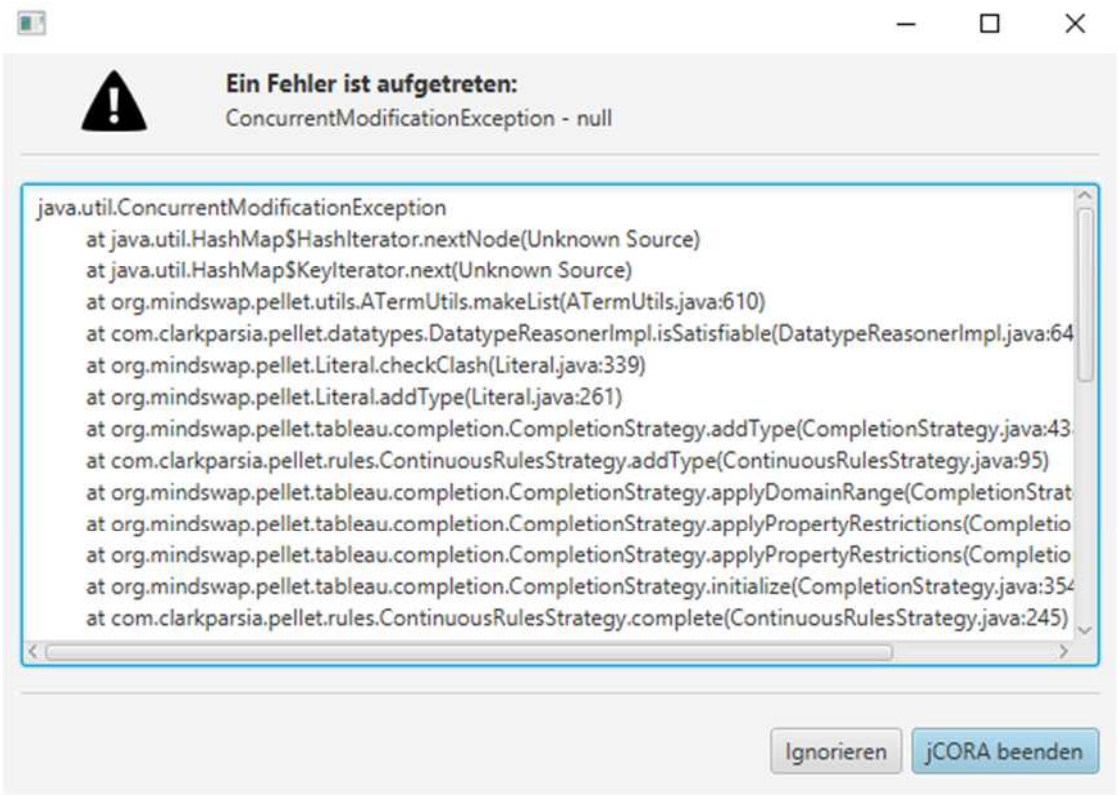


Abbildung 88: Exemplarische Fehlermeldung für die Ähnlichkeitsberechnung bei zu vielen Instanzen

Streng genommen ist aus der Fehlermeldung in Abbildung88 nicht ersichtlich, warum diese Fehlermeldung entstanden ist. Um eine Fehleranalyse durchzuführen, wäre es erforderlich, den Quellcode des CBR-Tools jCORA einzusehen. Die Autoren dieses Beitrags vermuten, dass diese Fehlermeldung im Zusammenhang mit der Anzahl und dem Umfang der Instanzen steht. Hinzu kommt, dass das CBR-Tool jCORA mit seiner Fallbasis (Projektwissensbasis) bei der Wissensrepräsentation über Fälle (Projekte) schnell sehr groß wird, sodass die Fallbasis allein bei der in diesem Beitrag durchgeführten Fallkonstruktion mehrere Gigabyte an Speicherplatz einnimmt. Ebenso ist der Verlust von Fällen möglich, wenn bei der Konstruktion von Instanzen Stringwerte einen größeren Bereich einnehmen, wie z. B. bei der Formulierung von zwei oder drei Sätzen.

Ein weiteres erwähnenswertes Problem bei der Fallkonstruktion ist der Verlust der Beschriftung der grafischen Darstellung mittels eines Fallgraphs. Dieser Beschriftungsverlust kann durch einen erneuten Aufruf des CBR-Tools jCORA behoben werden. Es lässt sich jedoch nicht unmittelbar nachvollziehen, warum dieser Fehler auftritt.

Grundsätzlich wäre eine Sammlung von bekannten Fehlern des CBR-Tools jCORA, ähnlich wie bei Protegé, wünschenswert, um über eine Liste von fehlerverursachenden Software-Problemen zu verfügen. Diese Probleme müssten in einer weiterführenden Problembetrachtung evaluiert und behoben werden. Oder die Erfahrungen mit den – insbesondere technischen – Problemen des CBR-Tools jCORA könnten für eine Weiterentwicklung der Software auf einer anderen technologischen Basis, z. B. als Cloud-native-Anwendung, genutzt werden, um nicht wieder in diese Probleme zu laufen.

In diesem Beitrag wurde auf eine vollständige technische Bewertung des CBR-Tools jCORA verzichtet, weil es primär nicht um eine Kritik der Probleme dieses CBR-Tools geht, sondern dessen „lösungsorientierte“ Weiterentwicklung im Vordergrund steht. Daher wird später in Kapitel 4 die „Machbarkeit“ einer Reimplementierung des CBR-Tools jCORA als eine Cloud-native-Anwendung vorgestellt, ebenso wie die Möglichkeit, weitere spezifische Ähnlichkeitsfunktionen ohne größere Anpassungen unter Verwendung Künstlicher Neuronaler Netze für die Ähnlichkeitsberechnung von Wörtern bereitzustellen.

3.3.2.2 Fall 1: Neuausrichtung eines Einsatzführungssystems der Polizei

Der hier betrachtete Fall trägt die Fall-ID „Neuausrichtung eines Einsatzführungssystems der Polizei“ und stellt exemplarisch einen Fall dar, der sich auf ein Vergabeverfahren (ausgedrückt durch die Instanz *AusschreibungEinsatzführungssystem*) auf die Lieferung eines Einsatzführungssystems erstreckt. Die nachfolgende Abbildung89 illustriert den gesamten Fallgraphen des Falls.

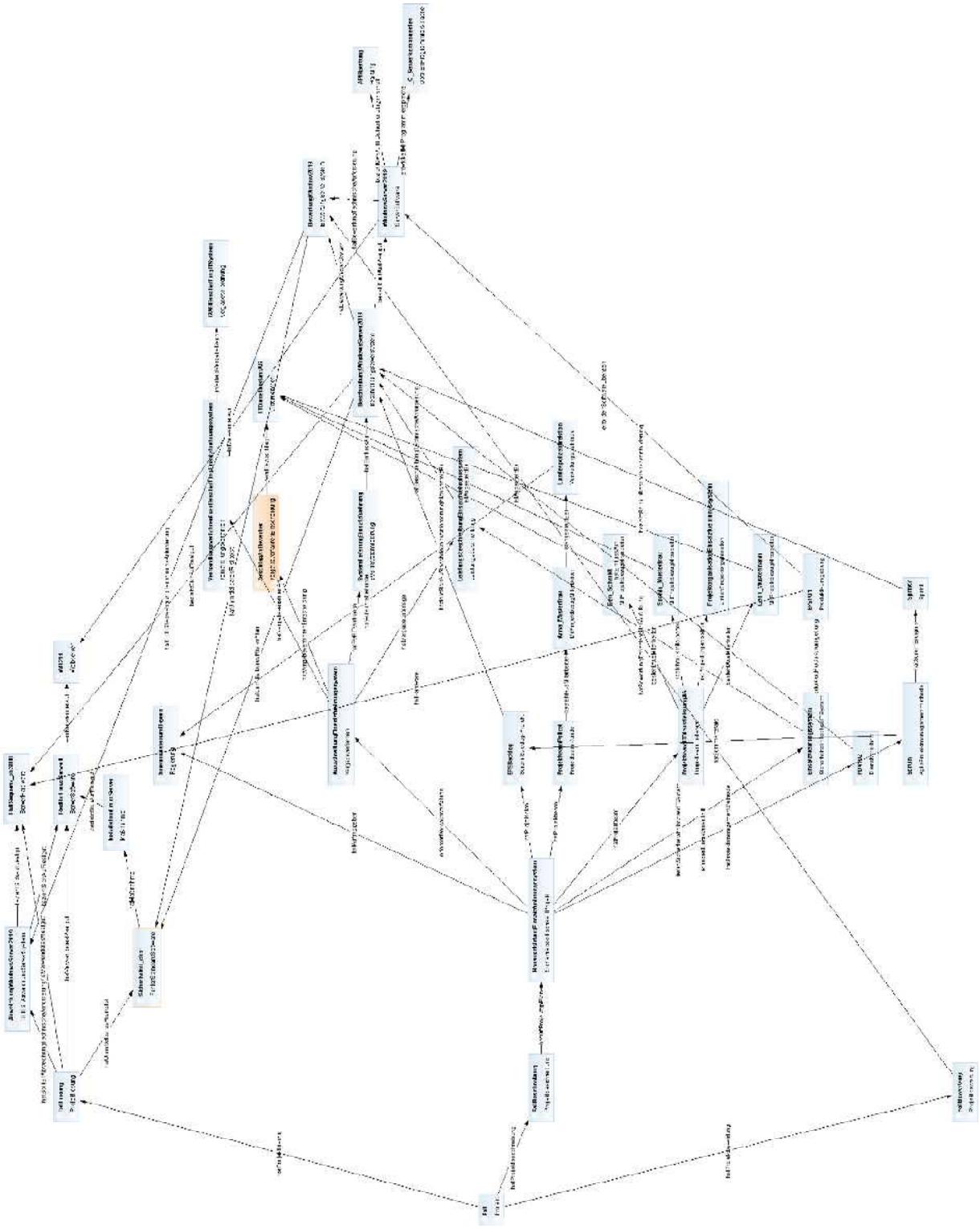


Abbildung 89: Fallgraph des Falls *Neuausrichtung eines Einsatzführungssystems der Polizei*

Das Vergabeverfahren wurde als Verhandlungsverfahren durchgeführt. Dies wird durch die Instanz *VerhandlungsverfahrenBeschaffungEinsatzführungssystem* ausgedrückt. Das Attribut *ist-europaweiteausschreibung* mit dem Wert *nein* gibt an, dass es sich nicht um eine europaweite Ausschreibung handelte. Den Zuschlag im Vergabeverfahren erhielt das Unternehmen IT-

Dienstleistung AG (Instanz *ITDienstleistungAG*), das als Attribute die *Mitarbeiteranzahl*, *Unternehmenssitz*, *Unternehmensform* und den *Unternehmensname* hat. Exemplarisch illustriert die nachfolgende Abbildung90 die Attribute der Instanz *ITDienstleistungAG* sowie die nicht-taxonomischen-Relationen, welche die Instanz *ITDienstleistungAG* in ihren Nachbereich besitzt.

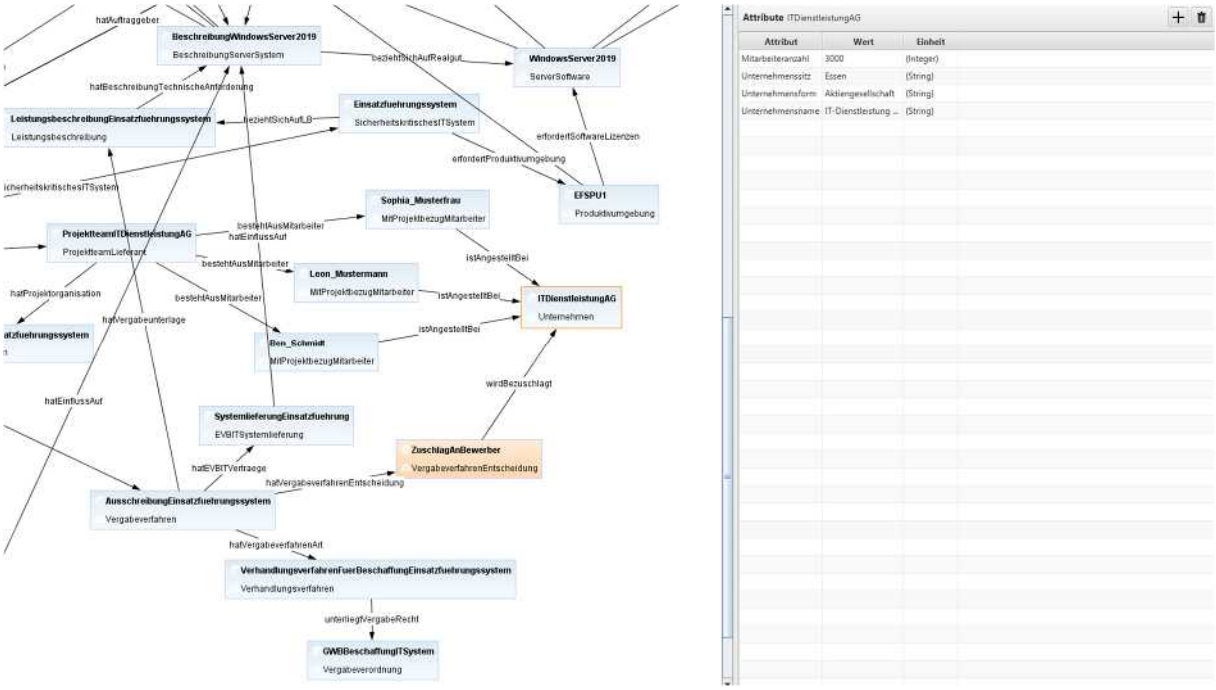


Abbildung 90: Attribute der Instanz *ITDienstleistungAG* und Ausschnitt des Fallgraphen

In der Leistungsbeschreibung des Vergabeverfahrens (Instanz: *LeistungsbeschreibungEinsatzführungssystem*) wird die Anforderung beschrieben, dass das Einsatzführungssystem über ein Server-System mit der Software Windows Server 2019 verfügen muss. Dies wird durch die nicht-taxonomische Relation *hatBeschreibungTechnischeAnforderung* beschrieben. Sie verbindet die Instanz *LeistungsbeschreibungEinsatzführungssystem* mit der Instanz *BeschreibungWindowsServer2019*. Um die Realgüter gemäß den Anforderungen auszudrücken, verbindet die nicht-taxonomische Relation *beziehtSichAufRealgut* die Instanz *BeschreibungWindowsServer2019* mit der Instanz *WindowsServer2019* und die Instanz *BeschreibungWindowsServer2019* mit der Instanz *BullSequana_xh30000*. Diese Konstruktion soll zum Ausdruck bringen, dass für die Umsetzung dieser Anforderung die Software Windows Server 2019 auf der Serverhardware „BullSequana“ bereitgestellt werden soll. Die beschriebene Anforderung des bereitzustellenden Windows Server 2019 besitzt ein geplantes und damit ein mögliches Risiko einer Sicherheitslücke, welches durch die Instanz *Sicherheitslücke* ausgedrückt wird. Diese Instanz wird durch die Instanz *BeschreibungWindowsServer2019* mit der nicht-taxonomischen Relation *hatUnmittelbaresRisikoPlan* verbunden. Als Maßnahme zur Behebung der Sicherheitslücke ist die Installation eines Linux Servers vorgesehen, die durch die nicht-taxonomische

Relation *hatMaßnahme* ausgedrückt wird. Die Instanz *InstallationLinuxServer* ist mit der Instanz *RedhatLinuxServer9* über die nicht-taxonomische Relation *beziehtSichAufRealgut* verbunden, wodurch ausgedrückt wird, dass ein anderes Realgut zur Lösung dieser Anforderung verwendet wird, falls diese Sicherheitslücke tatsächlich auftritt. Dass dieses Risiko der Sicherheitslücke tatsächlich eingetreten ist, zeigt die nicht-taxonomische Relation *hatUnmittelbares-RisikoIst.* Sie verbindet die Instanz *BeschreibungWindowsServer2019* mit der Instanz Sicherheitslücke. Die Instanz *SollIstAbweichungWindowsServer2019* geht auf die tatsächlichen Abweichungen in der Form ein, dass nicht-taxonomische Relationen von der Instanz zu den tatsächlich verwendeten Realgütern (hier *RedHatLinuxServer9* und *BullSequana*_xh30000) führen und im Attribut *hatErläuterungDerAbweichung* eine Erläuterung der Abweichung in Textform erfolgt. Die nachfolgende Abbildung91 illustriert den Ausschnitt der zuvor genannten Zusammenhänge.

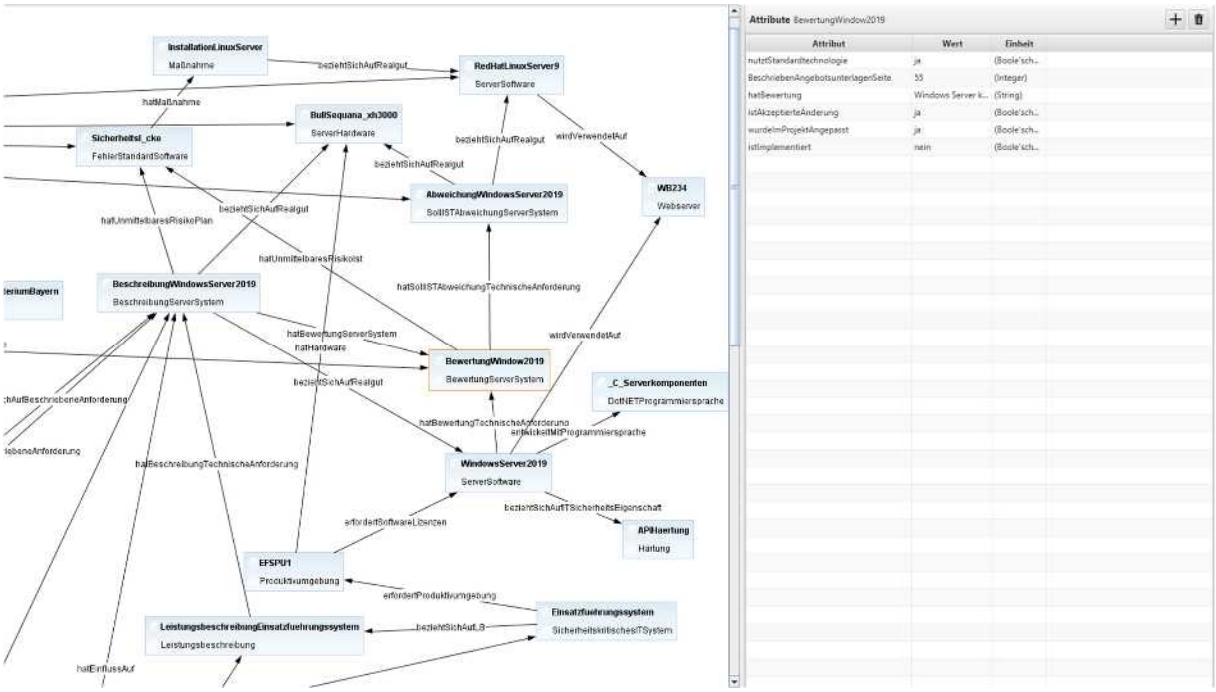


Abbildung 91: Anforderungen eines „Windows Server 2019“

Die Falllösung kann auf zwei Arten konstruiert werden. Eine erste Möglichkeit besteht darin, analog zur *Projektbeschreibung* vorzugehen. Hierbei wird anstelle der nicht-taxonomischen Relation *betrifftProjekttypPlan*, die die Instanzen *Fallbeschreibung* und *NeuausrichtungEinsatzführungssystem* (aus der Klasse *SicherheitskritischesITProjekt*) miteinander verbindet, die nicht-taxonomische Relation *betrifftProjekttypIst* verwendet. Diese Konstruktion erlaubt den Zugriff auf alle sprachlichen Ausdrucksmittel, die auch in der *Projektbeschreibung* möglich sind. Eine zweite Möglichkeit besteht darin, direkt auf die tatsächlich verwendeten Realgüter, Risiken und Abweichungsbeschreibungen zu verweisen, wie es z. B. in diesem Fall mittels nur in jCORA ausgelegter lokaler Instanzen geschehen ist.

Die nachfolgende Abbildung 92 zeigt beispielhaft, wie mit der nicht-taxonomischen Relation *hatVerwendetesRealgut* auf die tatsächlich verwendeten Realgüter und mit der nicht-taxonomischen Relation *hatTechnischeAnforderungAbweichung* auf die Abweichungen der technischen Anforderungen eingegangen wird. Ein analoges Vorgehen ist auch über weitere Relationen möglich, um z. B. auf die tatsächlichen Projektmitglieder oder Projektmanagementmethoden einzugehen.

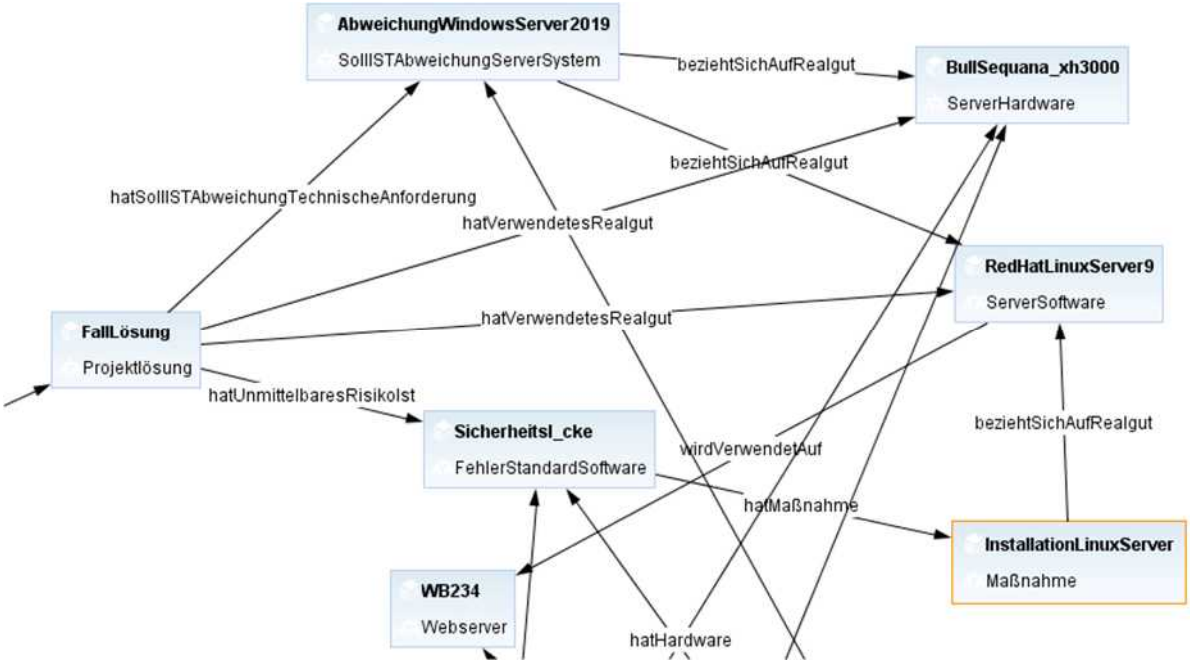


Abbildung 92: Falllösung exemplarisch für die Abweichung von der Anforderung „Windows Server 2019“

3.3.2.3 Fall 2: Aufbau einer Kooperativen Leitstelle

Der folgende Fall trägt die Fall-ID Aufbau „Kooperative Leitstelle“. Er stellt exemplarisch einen Fall dar, in dem in einem Vergabeverfahren (ausgedrückt durch die Instanz *Ausschreibung_Aufbau_Kooperative_Leitstelle*) die Lieferung eines Einsatzleitsystems gefordert wurde, welche sowohl von der Polizei als auch von der Feuerwehr genutzt werden soll. Die nachfolgende Abbildung93 illustriert den gesamten Fallgraphen des Falls.

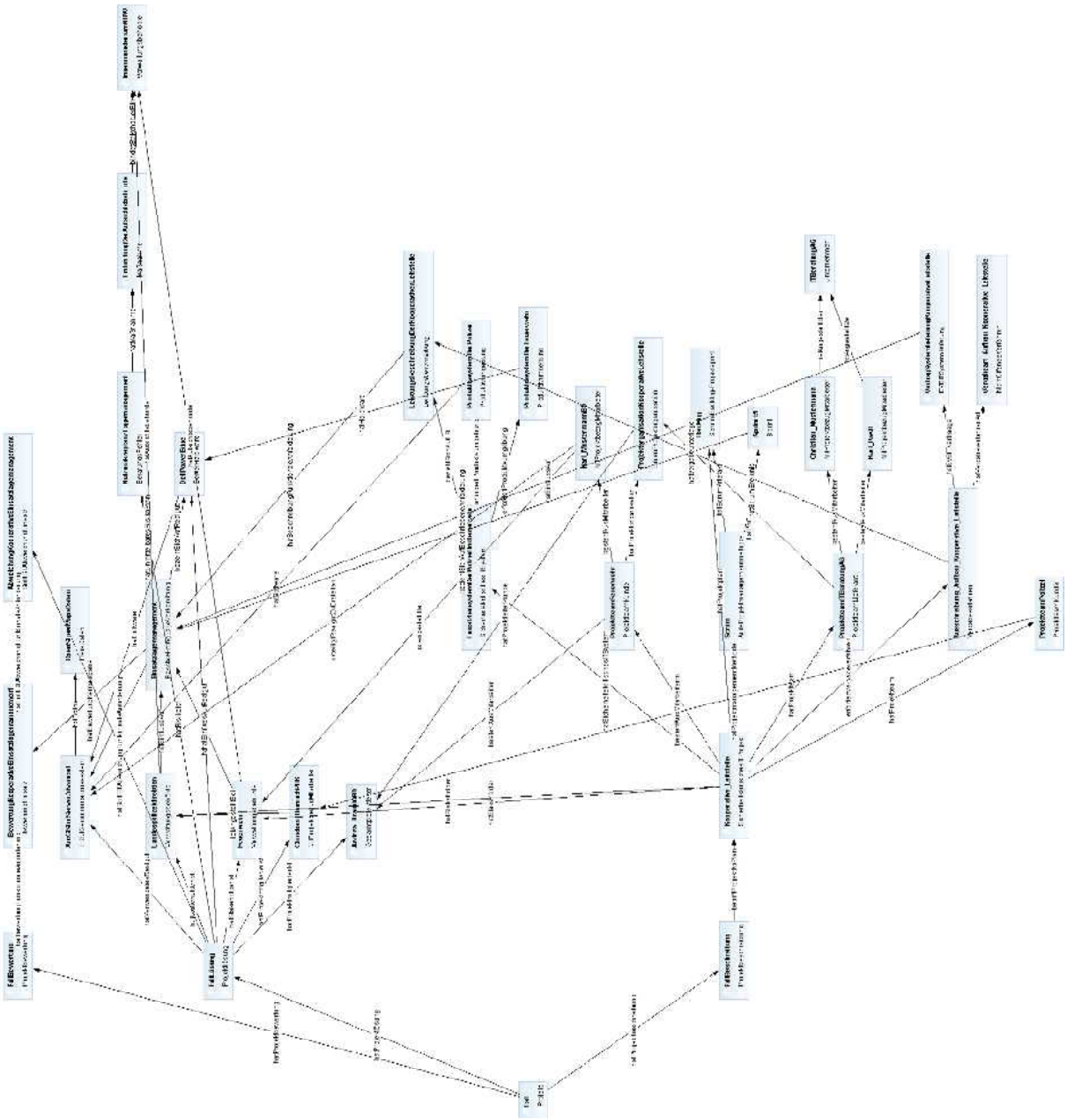


Abbildung 93: Fallgraph des Falls *Kooperative Leitstelle*

In der Leistungsbeschreibung, ausgedrückt als Instanz *LeistungsbeschreibungKooperativeLeitstelle*, wird eine Anforderung an das Lagemanagement, ausgedrückt als Instanz *Einsatzlagemangement*, beschrieben. Sowohl die Landespolizeidirektion als auch die Feuerwehr haben einen Einfluss auf die Anforderung, da beide die potenziellen Nutzer dieses Systems sind. Die Instanzen *Landespolizeidirektion* und *Feuerwehr* besitzen daher ein Attribut *hatDirektenBezugZuPotentiellemNutzer*, welches auf *ja* gesetzt ist. Diese Anforderung soll durch eine Standardsoftware umgesetzt werden, was mit der Instanz *ArcGISForServerAdvanced* ausgedrückt wird. Diese Software wird auf einer Hardware DellPowerEdge betrieben, die durch die Instanz *DellPowerEdge* ausgedrückt wird. Diese Hardware wird in der Produktivumgebung der Feuerwehr (*ProduktivumgebungFeuerwehr*) betrieben und greift auf frei verfügbare Geodaten namens Open Street Maps (*OpenStreetMapDaten*) zu. Das Einsatzleitsystem (*EinsatzleitsystemDerPolizeiUndFeuerwehr*) wird auf zwei Produktivumgebungen betrieben, eine bei der Polizei und eine bei der Feuerwehr. Sowohl die Polizei als auch die Feuerwehr greifen auf die Software zu. Diese wird jedoch nur auf der Hardware in der Produktivumgebung der Feuerwehr zur Verfügung gestellt, worauf auch die Polizei Zugriff hat. Diese Anforderung des Lagemanagements birgt das geplante Risiko, dass die Nutzerakzeptanz für das Lagemanagement fehlen könnte. Dieses Risiko ist in der Umsetzung nicht eingetreten, obwohl geplant war, bei Problemen mit der Nutzerakzeptanz einen Stakeholder, nämlich die Aufsichtsbehörde für Polizei und Feuerwehr (*InnenministeriumNRW*), einzubinden. Im Projekt wurden Abweichungen festgelegt, die durch die Instanz *AbweichungKoorperativesEinsatzmanagement* ausgedrückt wird. Beispielsweise wurde festgelegt, dass nicht alle Situationen mit einem gemeinsamen Lagemanagement behandelt werden müssen, sondern dass auch Ausnahmen definiert werden können. Ein interner Vermerk, ausgedrückt durch das Attribut *hatInternenVermerk*, das mit dem Datentyp *String* versehen wurde, beschreibt, was zum Kompromiss beigetragen hat, nämlich dass ein Änderungsmanagementverfahren angewendet wurde. Das Attribut *hatInternenVermerk* ist ein Beispiel dafür, dass, obwohl viele Zusammenhänge durch nicht-taxonomische Relationen ausgedrückt werden können, in der Praxis viele kleine Vermerke von großer Bedeutung sind, da sie Erfahrungswissen enthalten, das wiederverwendet werden kann. Dieses Feld, das einen String-Datentyp aufweist, kann z. B. später analysiert werden, um Ähnlichkeiten von Stringwerten durch spezifische Ähnlichkeitsfunktionen zu berechnen. Dazu kann die später implementierte Ähnlichkeitsfunktion für Stringwerte verwendet werden.

Die nachfolgende Abbildung94 illustriert den zuvor genannten Sachverhalt anhand eines Fallgraphen.

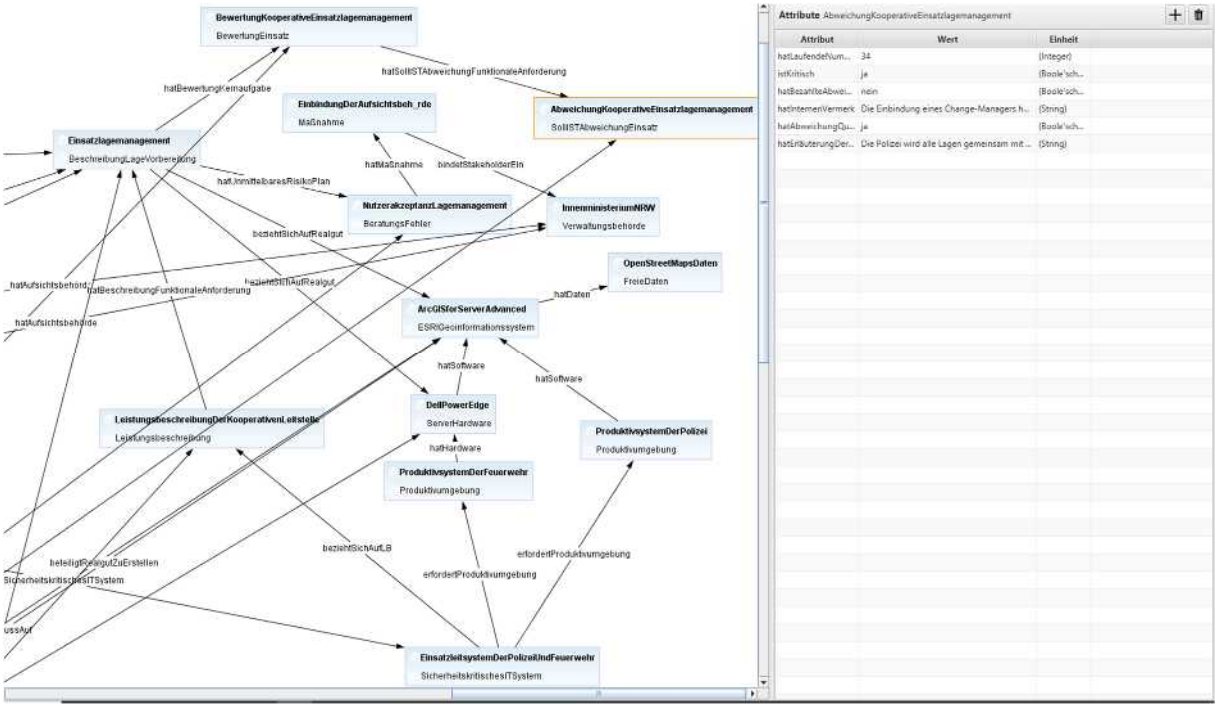


Abbildung 94: Anforderungen an das gemeinsame Lagemanagement

Die *Fallbewertung* besitzt nicht-taxonomische Relationen, wie *hatBewertungFunktionaleAnforderung*, die auf die Instanzen der Bewertung zugreifen können. Durch diese Modellierung kann direkt auf das Bewertungswissen der beschriebenen Anforderungen zugegriffen werden. Die Falllösung greift direkt auf die Wissenskomponenten zu, die zur konkreten Falllösung beigetragen haben, wie z. B. die tatsächlichen Stakeholder, Projektmitglieder, Realgüter und Abweichungen zwischen den Soll- und den Ist-Werten der beschriebenen Anforderungen. Die nachfolgende Abbildung95 veranschaulicht die voranstehenden Ausführungen.

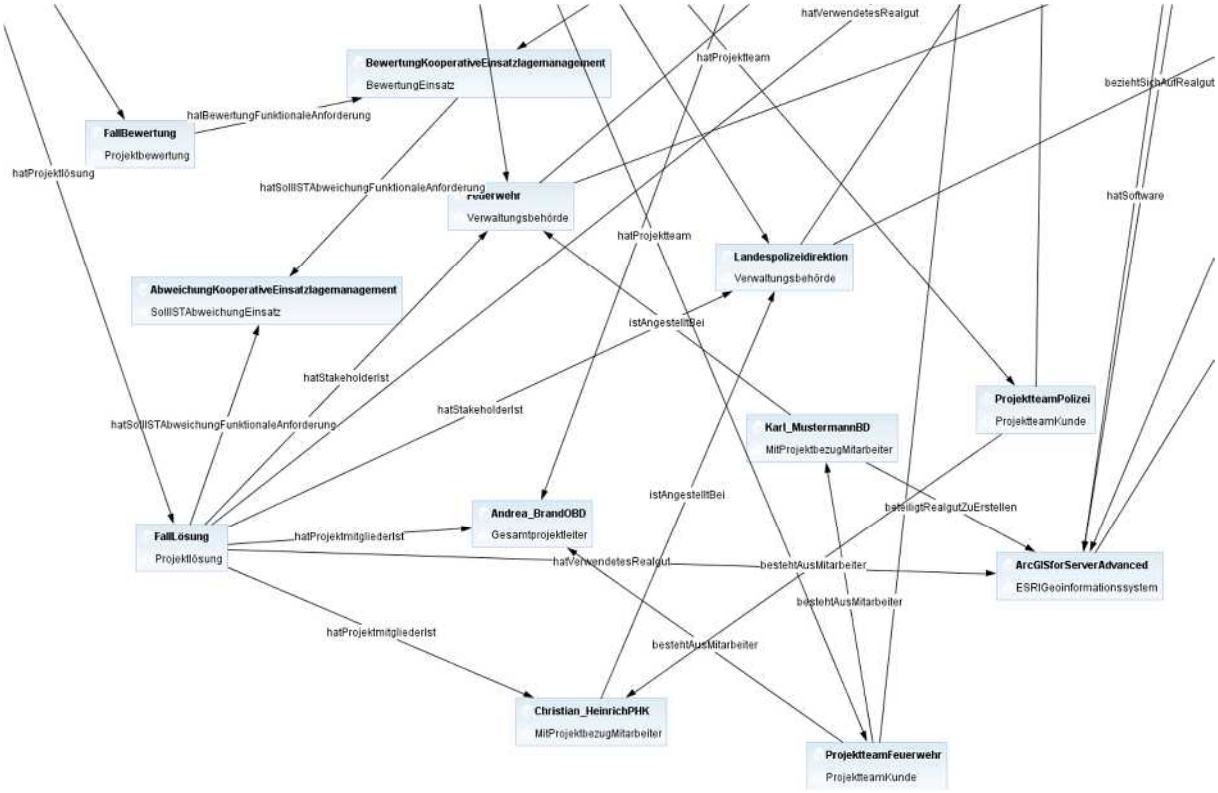


Abbildung 95: Exemplarischer Auszug der *Falllösung* und der *Fallbewertung*

3.3.2.4 Fall 3: Aufbau einer zentralen Datenbank für Ermittlungen

Der folgende Fall trägt die Fall-ID „Aufbau einer zentralen Datenbank für Ermittlungen“ und stellt exemplarisch einen Fall dar, in dem in einem Vergabeverfahren (ausgedrückt durch die Instanz *AusschreibungZentraleDatenbank*) die Lieferung einer zentralen Datenbank für Ermittlungsverfahren gefordert wurde, die von der Polizei in allen Bundesländern genutzt werden soll. Die nachfolgende Abbildung96 illustriert den gesamten Fallgraphen des Falls.

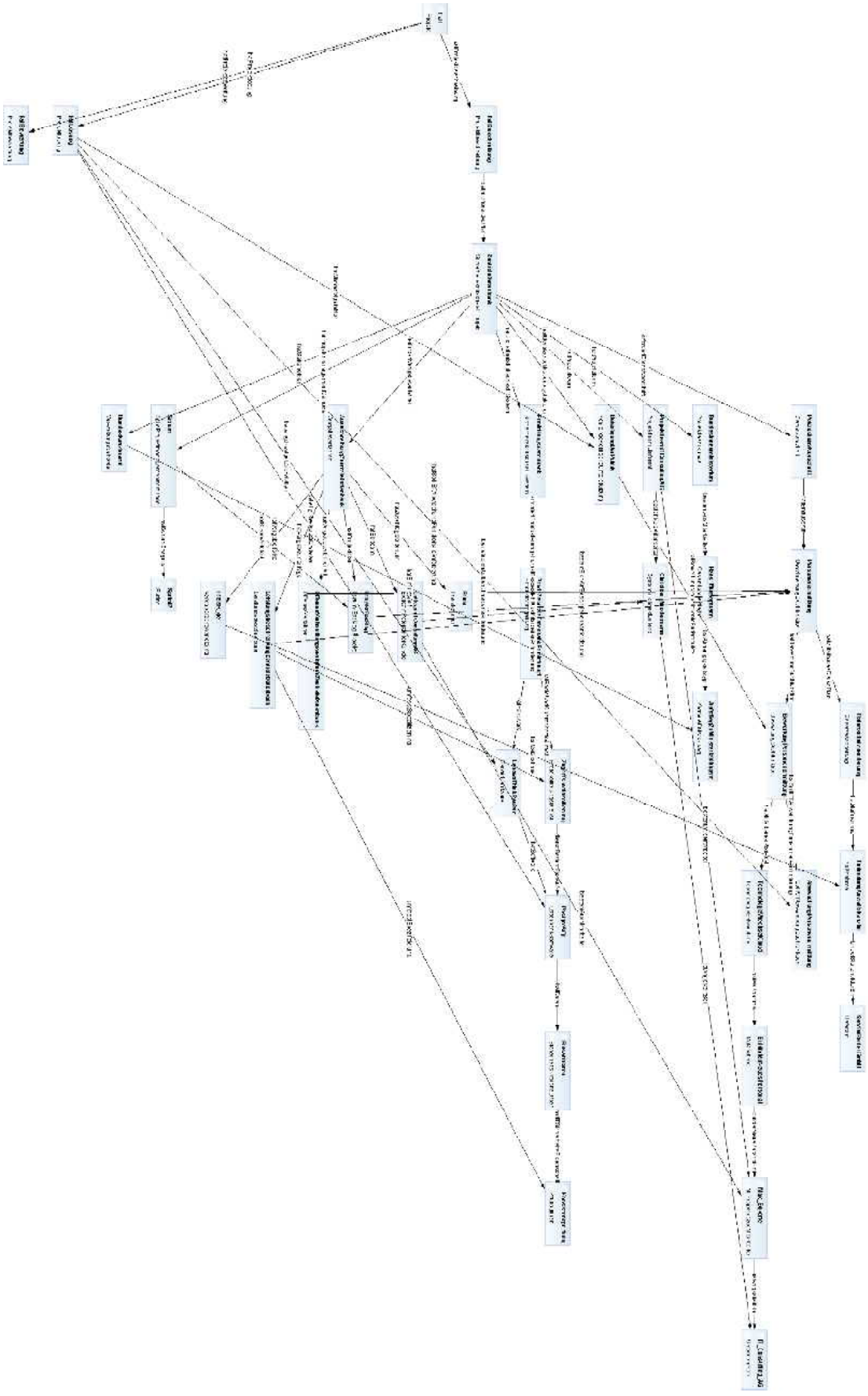


Abbildung 96: Fall *zentrale Datenbank für Ermittlungsverfahren*

Der Fall stellt z. B. dar, dass im Vergabeverfahren ein Vergaberisiko in der Form besteht, dass ein Vergaberechtsverstoß befürchtet wird, weil ausschließlich der Preis über den Zuschlag im Vergabeverfahren entschied. Die ausschließliche Entscheidung über den Zuschlag basierend auf dem Preis im Vergabeverfahren könnte als Vergaberechtsverstoß betrachtet werden, da dadurch potenziell Qualität, Innovation und das Prinzip der Bestenauswahl nach Wirtschaftlichkeit vernachlässigt werden könnten. Dies könnte zu minderwertigen Leistungen führen und einen wettbewerbsverzerrenden Effekt haben, der von einem Bieter angefochten werden könnte.

Das Zuschlagskriterium wird durch die Instanz *Preis* ausgedrückt. Die nicht-taxonomische Beziehung *hatZuschlagskriterium* verbindet die Instanz *AusschreibungZentraleDatenbank* mit der Instanz *Preis*. Um das Risiko eines Vergaberechtsverstoßes zu reduzieren, wurde eine Maßnahme implementiert, die durch die Instanz *EinbindungAnwaltskanzlei* ausgedrückt wird. Diese Maßnahme stellt dar, dass ein externer Vergaberechtsanwalt, ausgedrückt durch die Instanz *KanzleiReuterGmbH* hinzugezogen wird, um das Vergaberisiko zu reduzieren. Des Weiteren stellt der Fall dar, dass im Vergabeverfahren die gleiche Person beteiligt ist wie im Projekt. So ist sowohl die Instanz *BidteamITConsultingAG* mit der nicht-taxonomischen Relation *besteht-AusMitarbeiter* mit der Instanz *Christian_Mustermann* verknüpft als auch die Instanz *Projekt-teamITConsultingAG*. Dieser Zusammenhang ist insofern interessant, als Personen, die am Vergabeverfahren und auch am späteren Projekt beteiligt sind, einen geringeren Wissensverlust haben als Personen, die erst später in das Projekt einsteigen, weil die erstgenannten Personen von Anfang an am Projekt beteiligt sind. Es wäre denkbar, hier eine SWRL-Regel so zu gestalten, dass die kontinuierliche Anwesenheit einer Person aus dem Vergabeverfahren auch im späteren Projektverlauf einen Erfolgsfaktor darstellen kann.

Ein weiterer erwähnenswerter Sachverhalt in diesem Fall ist das Abhängigkeitsverhältnis des Gesamtprojektleiters des Kundens. So wird der Gesamtprojektleiter des Kunden dadurch motiviert, dass durch einen erfolgreichen Projektabschluss eine Beförderung zum Ministerialdirigenten erfolgen kann. Dies wird durch die Instanz *AufstiegZuMinisterialdirigent* ausgedrückt. Dazu wird die Instanz *Hans_Mustermann*, die den Gesamtprojektleiter repräsentiert, mit der Instanz *AufstiegZuMinisterialdirigent* über die nicht-taxonomische Relation *hatAbhängigkeits-typ* verbunden. Diese Beförderung wird von der Instanz *Bundeskanzleramt* beeinflusst. Die Instanz *ZentraleDatenbank*, die das sicherheitskritische IT-Projekt repräsentiert, ist über eine nicht-taxonomische Relation *hatStakeholder* mit der Instanz *Bundeskanzleramt* verbunden. Der Wert „*nein*“ für das Attribut *hatNegativenEinfluss* von der Instanz *AufstiegZumMinisterialdi-rigenten* drückt aus, dass dieser Abhängigkeitstyp keinen negativen Einfluss auf das Projekt hat. Das Attribut *hatAbhängigkeitstypErläuterung* enthält eine Erläuterung der Abhängigkeit. Der Datentyp ist *String*, sodass ganze Sätze zur Erläuterung der Abhängigkeit formuliert werden können.

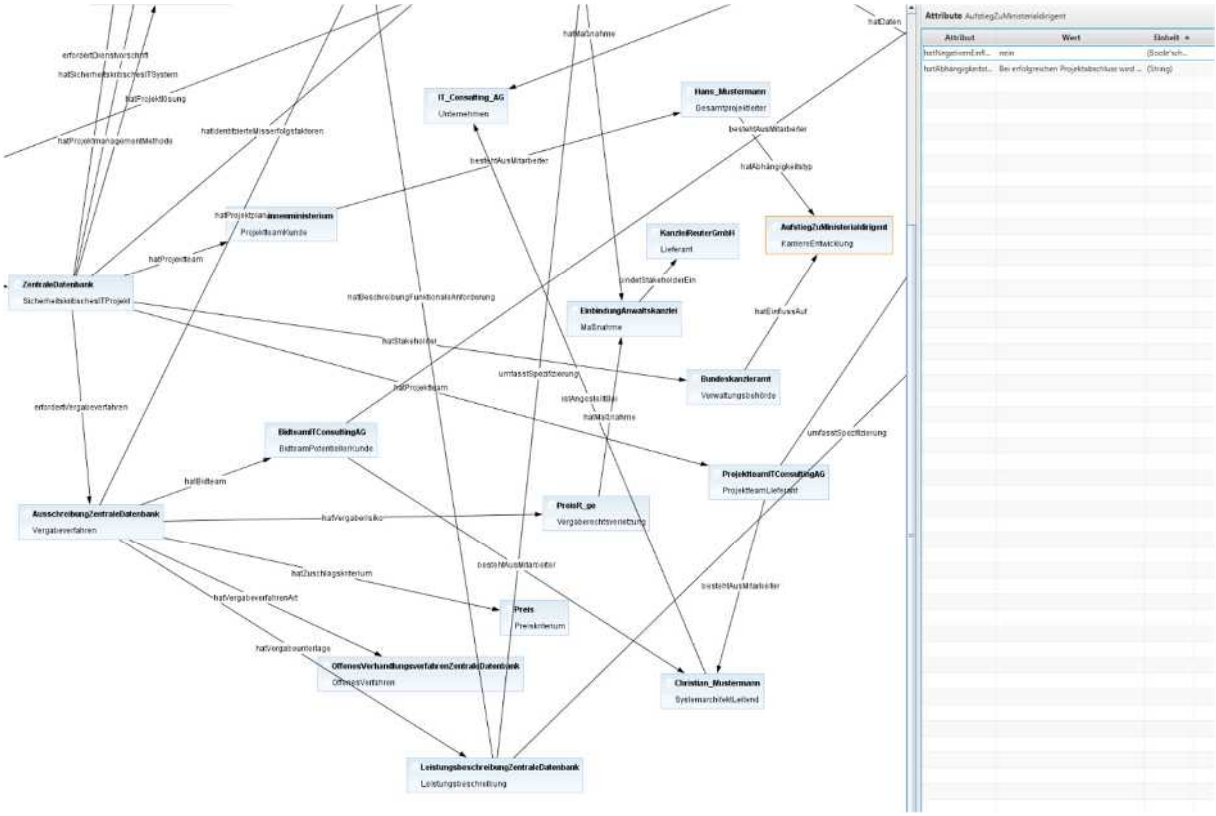


Abbildung 97: Darstellung des Projektteams

Darüber hinaus stellt der Fall eine Anforderung für eine Personenermittlung dar, die durch die gleichnamige Instanz *Personenermittlung* ausgedrückt wird. Das Attribut *hatBeschreibung*, das vom Datentyp *String* ist, beschreibt die Anforderung aus der Leistungsbeschreibung. Das Attribut *hatAngebotstext*, ebenfalls vom Datentyp *String*, enthält den konkreten Angebotstext zu dieser Anfrage. Die Instanz *Polizeidienstvorschrift*, die ebenfalls durch die gleichnamige Polizeidienstvorschrift ausgedrückt wird, hat einen Einfluss auf die Anforderung *Personenermittlung*. Es existiert auch eine Instanz *BeteiligungDerPolitik*, die Einfluss auf die Anforderung *Personenermittlung* hat, die über die nicht-taxonomische Relation *hatBeschreibungFunktionaleAnforderung* mit der Instanz *Personenermittlung* verknüpft ist. Dadurch wird ausgedrückt, dass dieses Projekt durch eine fehlende politische Unterstützung über einen Misserfolgsfaktor verfügt, was konkrete Auswirkung auf die Anforderung *Personenermittlung* hat. An dieser Stelle wird deutlich, dass eine klare Abgrenzung zwischen Risiken und Misserfolgsfaktoren eine Herausforderung darstellt. Misserfolgsfaktoren sind potenzielle Ursachen für das Eintreten von Risiken, welche den Erfolg eines Projektes beeinträchtigen oder sogar zum Scheitern eines Projektes führen können. Ein Beispiel dafür ist das Fehlen politischer Unterstützung. Weitere Beispiele für Misserfolgsfaktoren sind ein Mangel an Ressourcen und eine unzureichende Planung.

Die Implementierung der Anforderung *Personenermittlung* soll durch die Realgüter *PostgresSQL* und *LenovoThinkSystem* erfolgen. Eine Instanz *Zugriffsprotokollierung*, welche die Protokollierung der Zugriffe dokumentiert, bezieht sich auf das Realgut *PostgresSQL*.

In der ursprünglichen Planung wurde nur das Risiko identifiziert, dass eine Gesetzesänderung Auswirkungen auf die Anforderung der Personenermittlung hat. Dieses Risiko wird durch die Instanz *Datenschutzänderung* ausgedrückt. Eine nicht-taxonomische Relation *hatMittelbares-RisikoPlan* verbindet die Instanz *Personenermittlung* mit der Instanz *Datenschutzänderung*. Als Maßnahme ist die Einbindung eines externen Vergaberechtsanwalts vorgesehen, was aufgrund des Vergaberisikos bereits im Vergabeverfahren geschehen ist. Als tatsächliches Risiko ist jedoch eine Technologieentwicklung eingetreten, die mit der Instanz *TechnologieWechselCloud* ausgedrückt wird. Dies hatte zur Folge, dass neues Personal eingebunden werden musste. Die neu eingebundene Person wird durch die Instanz *Max_Expert* ausgedrückt. Die nachfolgende Abbildung98 veranschaulicht die zuvor erläuterten Punkte anhand eines Fallgraphen.

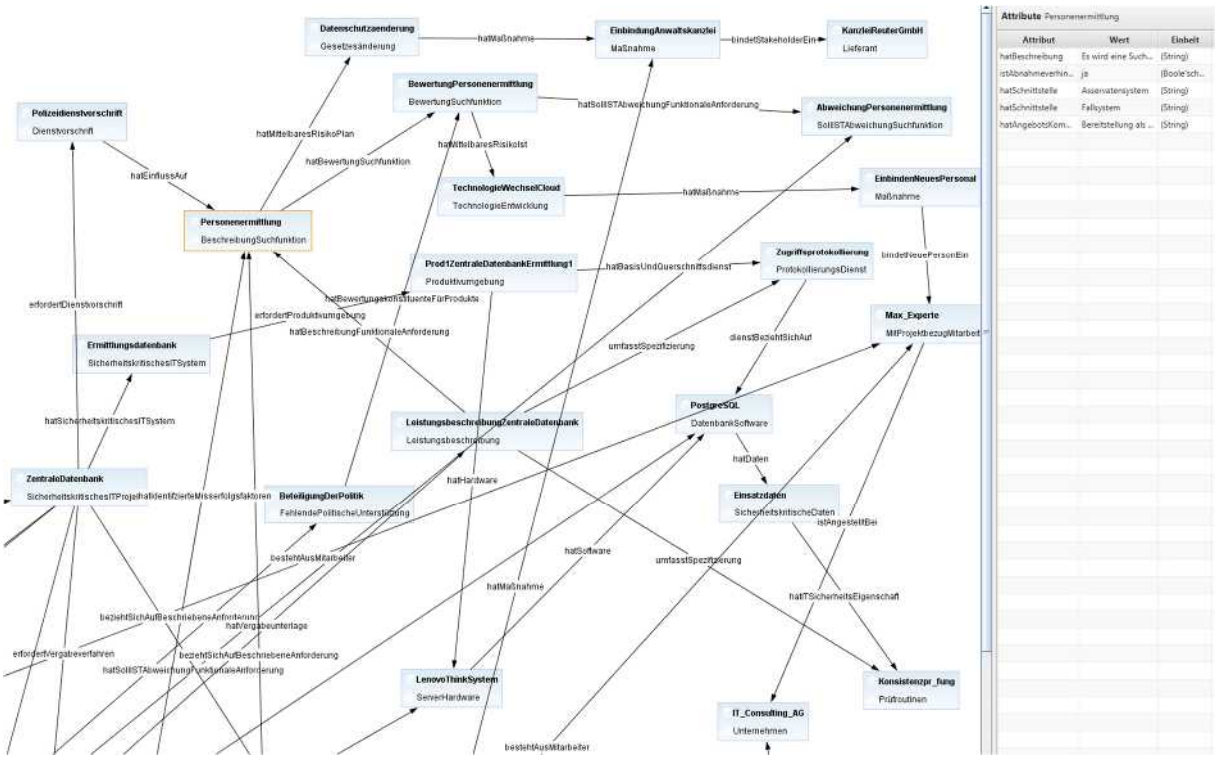


Abbildung 98: Darstellung einer Anforderung, eines Risikos und eines Misserfolgsfaktors

3.3.3 Ähnlichkeitsberechnungen

3.3.3.1 Berechnungsgrundlagen des CBR-Tools jCORA

Die Ähnlichkeitsberechnungen des CBR-Tools jCORA beruhen auf dem Ähnlichkeitsalgorithmus, der von BEIßEL (2011), S. 159-173, für ontologiegestützte CBR-Systeme entwickelt wurde. Dieser Ähnlichkeitsalgorithmus litt jedoch unter einigen „technischen“ Limitationen; vgl. BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 479-480. Sie führten dazu, dass sich der von BEIßEL entwickelte Algorithmus nicht in allgemeiner Weise zur Ähnlichkeitsberechnung

in ontologiegestützten CBR-Systemen einsetzen lässt. Daher wurde dieser Algorithmus in BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 492-511, zwecks allgemeiner Anwendbarkeit weiterentwickelt und im CBR-Tool jCORA implementiert. Auf diesen weiterentwickelten Ähnlichkeitsalgorithmus beziehen sich die nachfolgenden Ausführungen.

Für die Ähnlichkeitsberechnung im Rahmen des ontologiegestützten Case-based Reasonings stellt die Taxonomie einer Ontologie einen wichtigen Ausgangspunkt dar. Die gerichteten Pfade in einem Ontologiegraphen, die taxonomischen Subsumtions-Beziehungen zwischen den Klassen einer Ontologie wiedergeben, sowie die Länge dieser Pfade ermöglichen eine Berechnung der Ähnlichkeit mithilfe von Hilfsfunktionen aus der Graphentheorie. Darüber hinaus spielen für die Ähnlichkeitsberechnung aber auch die Attribute einer Klasse und die nicht-taxonomischen Relationen zwischen den Klassen eine wichtige Rolle. Diese Zusammenhänge werden im Folgenden anhand der zentralen Konstrukte der Konzeptähnlichkeiten sowie der partiellen und vollständigen Ähnlichkeiten näher erläutert. Dabei werden die Bezeichnungen „Klasse“ und „Konzept“ im Interesse der „Anschlussfähigkeit“ an die Fachliteratur synonym verwendet, weil zwar in diesem Beitrag der Klassenbegriff zunächst zugrunde gelegt wurde, aber im Kontext des hier vorgestellten Ähnlichkeitsalgorithmus bevorzugt von einer Konzeptähnlichkeit die Rede ist.

Die *Konzeptähnlichkeit* beschreibt die Ähnlichkeit von zwei Klassen (Konzepten) k_a, k_b, die zur selben Ontologie O gehören. Die Berechnung der Konzeptähnlichkeit erfolgt durch die Kombination von zwei Berechnungen:

- Berechnung der semantischen Distanzen zwischen den Klassen im Graphen der Ontologie (Ontologiegraph) sowie
- Berechnung der Ähnlichkeiten zwischen Klasseneigenschaften, die einerseits die Attribute einer Klasse umfassen und andererseits die Relationen, an denen eine Klasse partizipiert.

Zunächst wird die Berechnung der semantischen Distanzen erläutert. Dazu werden die nachfolgenden zwei Hilfsfunktionen benötigt, die auf der Graphentheorie aufbauen. Sie beziehen sich in beiden Fällen auf zwei beliebige Knoten des Ontologiegraphen, die jeweils eine Klasse repräsentieren. Diese beiden Hilfsfunktionen ermitteln:

- den Least Common Subsumer (LCS) als „niedrigsten“ Knoten, der im Ontologiegraphen zwei betrachteten Knoten gemeinsam übergeordnet ist, sowie
- die Länge des Pfads zwischen zwei betrachteten Knoten im Ontologiegraphen.

Die Hilfsfunktion *lcs* ermittelt für zwei Klassen k_a , k_b, die zum Ontologiegraphen einer Ontologie gehören, diejenige Klasse k_{lcs}, die sowohl die Subsumtions- als auch die Minimalitätseigenschaft des Least-Common-Subsumer-Klasse erfüllt. Vereinfacht ausgedrückt, berechnet die Hilfsfunktion *lcs* mit $lcs(k_a, k_b)$ diejenige Klasse k_{lcs} (LCS-Klasse), die erstens eine Oberklasse der Klassen k_a und k_b darstellt und zweitens die tiefstmögliche Position im Ontologiegraphen der Ontologie aufweist. Hierfür gilt:

$$lcs(k_a, k_b) = k_{lcs} \tag{1}$$

Die Hilfsfunktion *pfad* ermittelt für zwei Klassen k_a und k_b die kleinstmögliche Menge an Klassen, über die im Ontologiegraphen von der Klasse k_a aus die Klasse k_b auf einem zusammenhängenden Weg mit gleichsinnig gerichteten Kanten erreicht werden kann. Diese Hilfsfunktion lässt sich als minimale Weglänge zwischen den Klassen k_a und k_b im Ontologiegraphen auffassen, welche diejenigen Klassen aufzählt, die auf dem Weg mit minimaler Weglänge durchlaufen werden. Dabei werden die Klassen k_a und k_b zu Beginn und am Ende des Wegs im Ontologiegraphen mitgezählt. Hierfür gilt:

$$pfad(k_a, k_b) = \{k_a, k_n, k_m, \dots, k_b\} \tag{2}$$

Mithilfe der Hilfsfunktionen *lcs* und *pfad* wird die Funktion *dist* als semantische Distanz zwischen zwei Klassen berechnet. Die Funktion *dist* ermittelt die semantische Distanz als die maximale Länge der beiden Wege minimaler Länge, die sich zwischen den zwei hinsichtlich ihrer Ähnlichkeit zu vergleichenden Klassen k_a, k_b und der gemeinsam übergeordneten LCS-Klasse k_{lcs} erstrecken. Die semantische Distanz lässt sich als Kombination der Formeln (1) und (2) berechnen:

$$dist(k_a, k_b) = \max\left(\left|pfad\left(k_a, lcs(k_a, k_b)\right)\right|, \left|pfad\left(k_b, lcs(k_a, k_b)\right)\right|\right) \tag{3}$$

Als zweite Einflussgröße für die Berechnung der Konzeptähnlichkeit wird die Ähnlichkeit zwischen zwei Klassen k_a und k_b hinsichtlich ihrer *Klasseneigenschaften* – also ihrer *Attribute* und ihrer nicht-taxonomischen *Relationen* – betrachtet. Dazu werden zunächst die Klasseneigenschaften einer Klasse ermittelt. Hierfür dient die Hilfsfunktion $KE(k_a)$, die hier in exemplarischer Weise auf die Klasse k_a bezogen wird:

$$KE(k_a) = \{KE_1, KE_2 \dots .\} \tag{4}$$

Die Menge $KE(k_a)$ der Klasseneigenschaften einer Klasse k_a gibt an, welche Eigenschaften entweder durch explizite Spezifizierungen als Eigenschaften der Klasse k_a festgelegt sind oder indirekt durch die Oberklassen der Klasse k_a an diese Klasse im Sinne des objektorientierten Systemdesigns „vererbt“ werden. Mithilfe der Klasseneigenschaften lässt sich die Ähnlichkeit $sim_{ke}(k_a, k_b)$ zweier Klassen k_a und k_b hinsichtlich ihrer Eigenschaften wie folgt berechnen:

$$sim_{ke}(k_a, k_b) = \begin{cases} 0{,}0 \text{ wenn } KE(k_a) \cup KE(k_b) = \emptyset \\ \frac{|KE(k_a) \cap KE(k_b)|}{|KE(k_b) \cup KE(k_b)|} \text{ wenn } KE(k_a) \cup KE(k_b) \neq \emptyset \end{cases} \tag{5}$$

Die Ähnlichkeit zweier Klassen bezüglich ihrer Klasseneigenschaften wird also berechnet, indem die Anzahl der in der Schnittmenge festgestellten gemeinsamen Klasseneigenschaften durch die Anzahl aller Klasseneigenschaften der beiden miteinander zu vergleichenden Klassen dividiert wird. Sind jedoch beide Klasseneigenschaftsmengen leer, nimmt die Ähnlichkeit hinsichtlich der Klasseneigenschaften den Wert 0,0 an, weil es sinnlos anmutet, von einer (positiven) Ähnlichkeit in Bezug auf Klasseneigenschaften zu sprechen, wenn überhaupt keine Klasseneigenschaften vorliegen.

Durch die Kombination der semantischen Distanz mit der Berechnung der Ähnlichkeit bezüglich der Klasseneigenschaften lässt sich schließlich die Konzeptähnlichkeit $ksim(k_a, k_b)$ zwischen zwei Klassen k_a und k_b berechnen. Dazu wird zunächst die semantische Distanz der beiden Klassen k_a und k_b berechnet. Mithilfe des Reziprokwerts der semantischen Distanz ergibt sich die Ähnlichkeit der beiden Klassen auf Basis ihrer semantischen Distanz in der zugrunde liegenden Ontologie. Anschließend wird für die beiden Klassen k_a und k_b ihre Ähnlichkeit hinsichtlich der übereinstimmenden Klasseneigenschaften ermittelt. Die nachfolgende Formel (6) zeigt die Ermittlung der Konzeptähnlichkeit durch die Kombination der Formel (3) für die semantische Distanz mit der Formel (5) für die Berechnung der Ähnlichkeit hinsichtlich der Klasseneigenschaften:

$$ksim(k_a, k_b) = \begin{cases} 1{,}0 & wenn\ k_a = k_b \\ \frac{1}{dist(k_a, k_b)} * \frac{|KE(k_a) \cap KE(k_b)|}{|KE(k_a) \cup KE(k_b)|} & wenn\ k_a \neq k_b\ und\ KE(k_a) \cup KE(k_b) \neq \emptyset \\ 0{,}0 & wenn\ k_a \neq k_b\ und\ KE(k_a) \cup KE(k_b) = \emptyset \end{cases} \quad (6)$$

Die *Instanzenähnlichkeit* stellt eine Ähnlichkeitsberechnung dar, bei der zunächst die Konzeptähnlichkeit mit der Ähnlichkeit der für die Instanzen definierten Eigenschaften kombiniert wird. Das Ergebnis wird als *partielle* Instanzenähnlichkeit bezeichnet. Sie dient später als Grundlage für die Berechnung der *vollständigen* Instanzenähnlichkeit.

Zur Berechnung der Instanzenähnlichkeit dienen Ähnlichkeitstypen, die festlegen, auf welche Weise die Ähnlichkeit zwischen zwei Instanzen desselben Ähnlichkeitstyps ermittelt wird. Für jeden Ähnlichkeitstyp – repräsentiert durch den Parameter *n* – existiert in der Regel eine spezifische Ähnlichkeitsfunktion $sim_n(...)$. Zwar kann selten der Fall eintreten, dass für einen Ähnlichkeitstyp noch keine passende spezifische Ähnlichkeitsfunktion spezifiziert wurde. Für diesen Sonderfall kommen mehrere Optionen in Betracht. Beispielsweise kann der Ähnlichkeitswert 0 (für „vollständig unähnlich“) angenommen werden, wenn für den Ähnlichkeitstyp keine spezifische Ähnlichkeitsfunktion existiert. Dieser Sachverhalt wird in der nachfolgenden Betrachtung jedoch nicht berücksichtigt. Es ist zu betonen, dass spezifische Ähnlichkeitsfunktionen eine bedeutende Rolle für die Ähnlichkeitsberechnung spielen und dass sich die Verfügbarkeit einer solchen Funktion, die für einen bestimmten Ähnlichkeitstyp spezifiziert ist, als essenziell erweist, da andernfalls keine Ähnlichkeiten zwischen Instanzen mit diesem Ähnlichkeitstyp berechnet werden können.

Im Folgenden wird davon ausgegangen, dass für jede Instanzeigenschaft genau ein Ähnlichkeitstyp existiert, der genau eine spezifische Ähnlichkeitsfunktion besitzt. Daher wird eine 1:1:1-Kardinalität für die Instanzeigenschaften, Ähnlichkeitstypen und Ähnlichkeitsfunktionen unterstellt. An späterer Stelle werden in exemplarischer Weise spezifische Ähnlichkeitsfunktionen implementiert. Dabei wird die Funktion *simStringBOS* zur Ermittlung der Ähnlichkeit von Wörtern näher betrachtet.

Eine spezifische Ähnlichkeitsfunktion $sim_n(ie; a; b)$ ermittelt die Ähnlichkeit zwischen den Werten (Attribut- oder Relationswerten) *a* und *b* einer Instanzeigenschaft ie, die zum Ähnlichkeitstyp *n* gehört.

In Anlehnung an BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 502, DIVARI (2011), S. 25, STAAB (2011), S. 12, EL JERROUDI (2010), S. 40-41, und RICHTER (2008), S: 29-30, gelten für jede spezifische Ähnlichkeitsfunktion $sim_n(ie; a; b)$ folgende Funktionseigenschaften:

$$\text{Reflexivität: } sim_n(ie; a; a) = 1$$

$$\text{Symmetrie: } sim_n(ie; a; b) = sim_n(ie; b; a)$$

$$\text{Normierung: } sim_n(ie; a; b) \in [0; 1]$$

Für die Berechnung sowohl der partiellen Instanzenähnlichkeit als auch der vollständigen Instanzenähnlichkeit sind mehrere Symbole für Variablen, Hilfsfunktionen und Parameter erforderlich, die in der nachfolgenden Tabelle 55 beschrieben werden.

Symbol	**Beschreibung**
i_a	Die Variable i_a bezeichnet eine Instanz.
$IE(i_a)$	Die Hilfsfunktion *IE* ermittelt die Menge $IE(i_a)$ aller Eigenschaften der Instanz i_a.
n	Der Parameter n stellt einen Ähnlichkeitstyp dar, für den eine spezifische Ähnlichkeitsfunktion $sim_n(\dots)$ existiert.
$IE(i_a, n)$	Die Hilfsfunktion *IE* ermittelt die Menge $IE(i_a, n)$ aller Eigenschaften der Instanz i_a, die den Ähnlichkeitstyp n mit der spezifischen Ähnlichkeitsfunktion $sim_n(\dots)$ aufweisen.
D	Das Symbol D definiert die Menge aller Ähnlichkeitstypen. Somit gilt die Bedingung, dass $n \in D$ zutrifft.
ie	Die Variable ie beschreibt eine Eigenschaft einer Instanz. Die Instanzeigenschaft kann eine Relation oder ein Attribut sein.
w_{ie}	Die Variable w_{ie} gibt die Gewichtung der Instanzeigenschaft ie an.
$wert(i_a, ie)$	Die Hilfsfunktion $wert$ ermittelt für die Instanzeigenschaft ie die Menge $wert(i_a, ie)$ von Werten, die der Instanz i_a in Bezug auf diese Instanzeigenschaft zukommen.
IE	Die Menge IE definiert die Menge aller Instanzeigenschaften *ie*.
$IE_n\ (i_a, i_b)$	Die Menge IE_n enthält alle ähnlichkeitsrelevanten Instanzeigenschaften vom Ähnlichkeitstyp n, die den beiden Instanzen i_a und i_b gemeinsam zukommen.

$K(i_a)$	Die Hilfsfunktion K ermittelt die Klasse $K(i_a)$ der Instanz i_a.

Tabelle 55: Variablen, Hilfsfunktionen und Parameter für die Berechnung der Instanzenähnlichkeiten

Auf Basis der in der Tabelle55 erläuterten Variablen, Hilfsfunktionen und Parameter lässt sich zunächst die *partielle Instanzenähnlichkeit* ermitteln. Zu diesem Zweck werden für die zu vergleichenden Instanzen i_a und i_b alle ihre gemeinsamen Instanzeigenschaften vom Ähnlichkeitstyp n ermittelt:

$$IE_n\ (i_a, i_b) = \{IE(i_a, n) \cap IE(i_b, n)\} \text{ mit } n\ \in D \tag{7}$$

Sollte für diese Menge gemeinsamer Instanzeigenschaften $IE_n\ (i_a, i_b) = \emptyset$ gelten, ist die Instanzenähnlichkeit hinsichtlich des Ähnlichkeitstyps n gleich 0.

Die Ähnlichkeit zwischen zwei Instanzen i_a und i_b wird im Hinblick auf alle Ähnlichkeitstypen n und alle gemeinsamen Instanzeigenschaften mittels der nachfolgenden eigenschaftsbezogenen *partiellen* Ähnlichkeitsfunktion $esim(i_a, i_b)$ berechnet:

$$esim(i_a, i_b) = \sum_{n=1}^{D} \begin{cases} \sum_{ie\ \in\ (IE(i_a,n)\ \cap IE(i_b,n))} w_{ie} * h(n, ie, a, b) & \text{für } (IE(i_a, n)\ \cap IE(i_b, n)) \neq\ \emptyset \\ 0 & \text{für } \left(IE(i_a, n)\ \cap IE(i_b, n)\right) =\ \emptyset \end{cases}$$

mit: (8)

$$h(n, ie, a, b) = \frac{\sum_{a \in wert(i_a, ie)} \left(\underbrace{max}_{b \in wert(i_b, ie)} sim_n(ie, a, b) \right) + \sum_{b \in wert(i_b, ie)} \left(\underbrace{max}_{a \in wert(i_a, ie)} sim_n(ie, b, a) \right)}{|wert(i_a, ie)| + |wert(i_b, ie)|}$$

Gemäß Formel 8 wird für jeden Ähnlichkeitstyp n und für jede zugehörige Instanzeigenschaft ie aus der Menge IE_n einerseits für jedes Element a aus der Menge $wert(i_a, ie)$ der für die Instanz i_a definierten Werte der Instanzeigenschaft ie die maximale Ähnlichkeit $sim_n(ie, a, b)$ bezüglich jedes Elements b aus der Menge $wert(i_b, ie)$ der für die Instanz i_b definierten Werte der Instanzeigenschaft ie mithilfe der spezifischen Ähnlichkeitsfunktion sim_n des Typs n ermittelt. Andererseits wird für jedes Element b aus der Menge $wert(i_b, ie)$ der für die Instanz i_b definierten Werte der Instanzeigenschaft ie die maximale Ähnlichkeit $sim_n(ie, b, a)$ bezüglich jedes Elements a aus der Menge $wert(i_a, ie)$ der für die Instanz i_a definierten Werte der Instanzeigenschaft ie ebenso mithilfe der spezifischen Ähnlichkeitsfunktion sim_n des Typs n ermittelt. Vereinfacht ausgedrückt, wird jeder Wert a der Instanzeigenschaft ie für die Instanz i_a sequenziell mit den Werten b der Instanzeigenschaft ie für die Instanz i_b verglichen. Dabei

wird der maximale Ähnlichkeitswert gemäß den Werten $sim_n(ie, a, b)$ der spezifischen Ähnlichkeitsfunktionen sim_n für die Ähnlichkeitstypen *n* ausgewählt. Anschließend wird auf analoge Weise durch den Tausch der Instanzen i_a und i_b vorgegangen. Die ermittelten Summen für alle Werte *a* und *b* der Instanzeigenschaft ie werden durch die Anzahlen aller Instanzeigenschaftswerte *a* und *b* aus den Mengen $wert(i_a, ie)$ bzw. $wert(i_b, ie)$ dividiert; vgl. BERGENRODT/KOWALSKI/ZELEWSKI (2015), S. 505. Durch diese Normierung der zuvor berechneten Summen wird eine durchschnittliche Ähnlichkeit für jede Instanzeigenschaft ie ermittelt und abschließend mit dem Gewicht w_{ie} multipliziert, das der Instanzeigenschaft ie aus der Sicht der Benutzer des CBR-Tools jCORA zukommt.

Mithilfe der zuvor erläuterten partiellen Instanzenähnlichkeit und der früher vorgestellten Konzeptähnlichkeit kann die Instanzenähnlichkeit für zwei Instanzen i_a und i_b vollständig berechnet werden. Die *vollständige Instanzenähnlichkeit* für zwei Instanzen i_a und i_b lässt sich mithilfe der Funktion $isim$ ermitteln, indem die beiden bereits vorgestellten Ähnlichkeitsberechnungen hinsichtlich der Konzeptähnlichkeit sowie der partiellen Instanzenähnlichkeit multiplikativ miteinander verknüpft werden. Zusätzlich wird die partielle Instanzenähnlichkeit durch die Summe der Gewichte aller Eigenschaften ie der Instanzen i_a und i_b normiert. Daher gilt für die Funktion $isim$ der vollständigen Instanzenähnlichkeit für den Normalfall, dass die Mengen der Instanzeigenschaften für die beiden miteinander verglichenen Instanzen i_a und i_b jeweils nicht leer sind und beide Instanzen i_a und i_b mindestens eine gemeinsame Instanzeigenschaft besitzen:

$$isim(i_a, i_b) = ksim\big(K(i_a), K(i_b)\big) * \frac{esim(i_a, i_b)}{\sum_{ie \in \left(IE(i_a) \cup IE(i_b)\right)} w_{ie}} \tag{9}$$

$$\text{wenn } IE(i_a) \neq \emptyset \text{ und } IE(i_b) \neq \emptyset \text{ und } IE(i_a) \cup IE(i_b) \neq \emptyset$$

Für die Berechnung der Instanzenähnlichkeit sind zwei Sonderfälle zu berücksichtigen.

Der erste Sonderfall tritt ein, wenn für die beiden Instanzen i_a und i_b keine Instanzeigenschaften definiert sind. Die Vereinigungsmenge beider Instanzeigenschaftsmengen $IE(i_a)$ und $IE(i_b)$ wäre in diesem Sonderfall leer. Für diesen ersten Sonderfall wird die normierte partielle Ähnlichkeit zwischen den Instanzen i_a und i_b – also der zweite Faktor im Produkt der Formel 9 – auf den Wert 1 festgelegt:

$$isim(i_a, i_b) = \begin{cases} ksim(K(i_a), K(i_b)) * 1 = ksim(K(i_a), K(i_b)) \\ \text{wenn } IE(i_a) \cup IE(i_b) = \emptyset \end{cases} \tag{10}$$

Der zweite Sonderfall tritt ein, wenn zwar die Vereinigungsmenge der beiden Instanzeigenschaftsmengen $IE(i_a)$ und $IE(i_b)$ nicht leer ist, aber für eine der beiden Instanzen i_a oder i_b keine Instanzeigenschaften definiert sind. Für diesen zweiten Sonderfall wird die normierte partielle Ähnlichkeit zwischen den Instanzen i_a und i_b – also der zweite Faktor im Produkt der Formel 9 – auf den Wert 0 festgelegt:

$$isim(i_a, i_b) = \begin{cases} ksim(K(i_a), K(i_b)) * 0 = 0 \\ \text{wenn } IE(i_a) \cup IE(i_b) \neq \emptyset \text{ und } (IE(i_a) = \emptyset \text{ oder } IE(i_b) = \emptyset) \end{cases} \quad (11)$$

3.3.3.2 Exemplarische Ähnlichkeitsberechnung mithilfe des CBR-Tools jCORA

Exemplarisch erfolgt eine „manuelle“ Ähnlichkeitsberechnung für die beiden Projekte *RegionalleitstellenverbundSchleswigHolstein* und *KooperativeLeitstelleBerlin*. Diese beiden Projekte gehören als Instanzen zu den Klassen *SicherheitskritischesITProjekt* bzw. *Vergabeverfahren* aus der sicherheitskritischen IT-Projekt-Ontologie. Die Ähnlichkeitsberechnung ist daher – wie bei ontologiegestützten CBR-Systemen üblich – primär auf der Instanzenebene der Ontologie einschließlich der Instanzeigenschaften angesiedelt, greift aber sekundär auch auf die Klassen der Ontologie mit ihren Klasseneigenschaften zurück.

Um die Ähnlichkeitsberechnung übersichtlich durchführen zu können, wird der nachfolgende vereinfachte Ausschnitt aus der sicherheitskritischen IT-Projekt-Ontologie betrachtet.

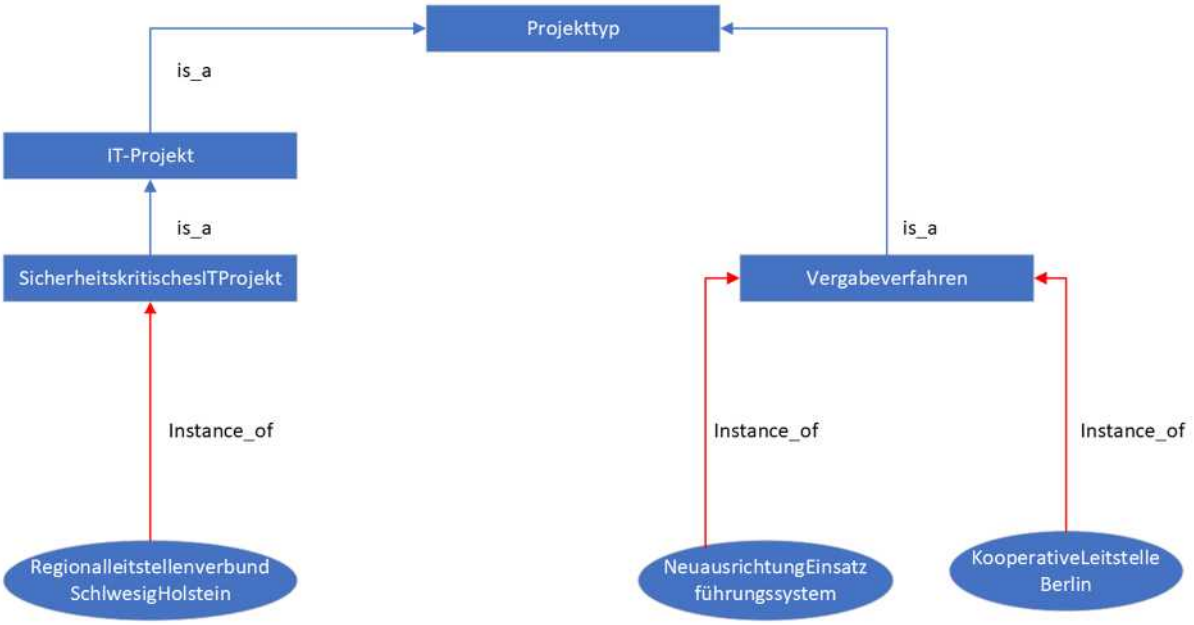


Abbildung 99: Ausschnitt aus der sicherheitskritischen IT-Projekt-Ontologie

Die jeweils vier Instanzeigenschaften der drei Instanzen in der voranstehenden Abbildung 99 sind in der nachfolgenden Abbildung100 zu sehen. Zusätzlich werden die konkreten Werte der Instanzeigenschaften angezeigt.

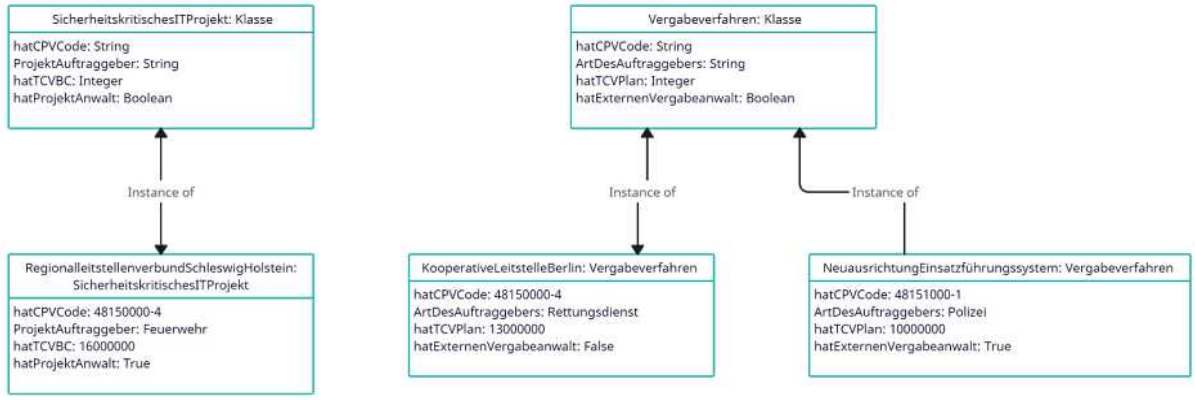


Abbildung 100: Darstellung der Werte der Instanzeigenschaften

Zunächst wird die Konzeptähnlichkeit ($ksim$) berechnet, anschließend erfolgt die Berechnung der partiellen Instanzenähnlichkeit ($esim$) und zuletzt wird die vollständige Instanzenähnlichkeit ($isim$) für die beiden Instanzen (Projekte) *RegionalleitstellenverbundSchleswigHolstein* und *KooperativeLeitstelleBerlin* berechnet.

Im Folgenden werden die Ergebnisse der Ähnlichkeitsberechnungen mithilfe des CBR-Tools jCORA auf Basis der drei Fälle dargestellt, die in den Kapiteln 3.3.2.2 bis 3.3.2.4 vorgestellt wurden. Sie betreffen den Fall 1 *Neuausrichtung_eines_Einsatzführungssystems_der_Polizei* als, den Fall 2 *Aufbau_Kooperative_Leitstelle* sowie den Fall 3 *Aufbau_einer_zentralen_Datenbank_für_Ermittlungen*. Auf den Ähnlichkeitsalgorithmus wird im Folgenden nicht eingegangen, weil er bereits in Kapitel 3.3.3.1 näher erläutert wurde.

Die Ähnlichkeitsberechnungen basieren auf dem Fall 2 *Aufbau_Kooperative_Leitstelle*. Die Berechnung der Ähnlichkeiten dauert 1 Minute und 13 Sekunden. Bei einer Ähnlichkeitsberechnung von mehr als einer Minute für drei Fälle, die nicht über das gesamte Fallwissen des jeweils betroffenen Projekts verfügen, ist davon auszugehen, dass sich die Laufzeit bei vollständigem Fallwissen und deutlich mehr als drei Fällen in der Fallbasis wesentlich erhöhen wird. Das könnte in der betrieblichen Praxis zu Akzeptanzverlusten gegenüber dem CBR-Tool jCORA führen.

Das Ergebnis in Bezug auf den Fall 2 *Aufbau_Kooperative_Leitstelle* sieht wie folgt aus:

Fall-ID	Ähnlichkeit		Adaptieren	Anzeigen
Aufbau_Kooperative_Leitstelle	100%		Adaptieren	Anzeigen
Neuausrichtung_eines_Einsatzf_hrungssy...	36%		Adaptieren	Anzeigen
Aufbau_einer_zentralen_Datenbank_f_r_E...	29%		Adaptieren	Anzeigen

Abbildung 101: Ähnlichkeitsberechnung für die Fälle 1 bis 3 mithilfe von jCORA

Da in jCORA spezifische Ähnlichkeitsfunktionen fehlen, müssen die berechneten Ähnlichkeitswerte kritisch beurteilt werden. Jedoch gibt die berechnete Ähnlichkeit eine erste Tendenz, um eine Ähnlichkeit zwischen den drei sicherheitskritischen IT-Projekten ableiten zu können.

Es ist richtig, dass die sicherheitskritischen IT-Projekte *Aufbau_Kooperative_Leitstelle* (Fall 2) und *Neuausrichtung_eines_Einsatzführungssystems_der_Polizei* (Fall 1) als ähnlicher eingestuft werden als die sicherheitskritischen IT-Projekte *Aufbau_Kooperative_Leitstelle* (Fall 2) und *Aufbau_einer_zentralen_Datenbank_für_Ermittlungen* (Fall 3).

Die Ähnlichkeit zwischen Fall 2 und Fall 1 wird mit 36 % angegeben. Die Richtigkeit des Ähnlichkeitswerts wird damit begründet, dass Fall 2 und Fall 1 beide Einsatzleitsysteme als Liefergegenstand vorsehen und sich neben dem Vergabeverfahren auch die Anforderungen ähneln. In Fall 2 wird ein kooperativer Ansatz zwischen Polizei und Feuerwehr verfolgt, der im Fall 1 nicht vorhanden ist, da das System ausschließlich von der Polizei genutzt wird. Daher mutet die Einschätzung, dass die beiden Fälle aufgrund dieser genannten Unterschiede keine sehr hohe Ähnlichkeit aufweisen, plausibel an.

Die Ähnlichkeit zwischen Fall 2 und Fall 3 wird mit 29 % angegeben, was ebenfalls als plausibel betrachtet wird, da sowohl die Projektmanagementmethode als auch die grundlegende Struktur des Vergabeverfahrens Ähnlichkeiten aufweisen. Auch hier kann von einer gewissen Ähnlichkeit ausgegangen werden, jedoch nicht in dem Maße wie zwischen Fall 2 und Fall 1, bezüglich derer die Ähnlichkeit mit 36 % berechnet wird.

Auch wenn die Ähnlichkeitswerte als tendenziell richtig anzusehen sind, erscheinen die konkreten Ähnlichkeitswerte mit 36 % bzw. 29 % als zu gering. Die niedrige Ähnlichkeit könnte auf folgende Gründe zurückzuführen sein, wie zum Beispiel die unvollständige Falldarstellung. Die präsentierten Fälle beinhalten möglicherweise nicht das gesamte relevante Fallwissen, was zu verringerten Ähnlichkeitswerten führen könnte, da einige Gemeinsamkeiten nicht erfasst wurden. Zusätzlich wurden alle Instanzeigenschaften gleichgewichtet. Die Entscheidung zur Gleichgewichtung aller Instanzeigenschaften wird dadurch begründet, dass in den nachfolgenden Ausführungen die Ähnlichkeitsberechnung behandelt wird. Eine gleichmäßige Gewichtung bietet eine robuste Grundlage, um die Vergleiche zwischen den Fällen zu ermöglichen, wodurch die Vergleichbarkeit von Fällen unabhängig von ihren spezifischen Instanzeigenschaften gewährleistet wird. Dies erleichtert zwar die Interpretation und Kommunikation der Ergebnisse, aber es spiegelt die Realität unzureichend wider. Diese Diskrepanz wird bewusst in Kauf genommen, da die grundsätzliche Durchführbarkeit einer Ähnlichkeitsberechnung unter der Verwendung des CBR-Tools jCORA im Vordergrund steht.

4 Konzipierung eines ontologiegestützten Case-based-Reasoning-Systems als Cloud-native-Anwendung

4.1 Vorbemerkungen zur Systemkonzipierung

Es wird ein konzeptioneller Ansatz vorgestellt, wie ein ontologiegestütztes CBR-System in einer Cloud-Umgebung implementiert werden kann. Dazu werden exemplarisch ausgewählte Funktionen als Serverless-Funktionen implementiert. Für die Implementierung der Serverless-Funktionen werden die Entwicklungsumgebungen Cloud9 der Firma Amazon Web Service sowie Google Colab der Firma Google genutzt. AWS Cloud9 sowie Google Colab sind Cloud-basierte integrierte Entwicklungsumgebungen, die es ermöglichen, ausschließlich im Browser den Code zu schreiben, auszuführen und zu debuggen. Für eine weitergehende Erläuterung der Entwicklungsumgebung Cloud9 wird auf AMAZON WEB SERVICES, INC. (2022b) verwiesen, für die Erläuterung der Entwicklungsumgebung Google Colab auf GOOGLE (2022).

Grundsätzlich stehen verschiedene Ansätze zur Verfügung, um eine Software („Anwendung“) wie das CBR-Tool jCORA in einer Cloud-Umgebung bereitzustellen. Die gängigsten Vorgehensweisen für die Überführung von bestehenden monolithischen Anwendungen bestehen darin, eine Anwendung auf die Cloud zu übertragen (Cloud-enabling) oder gänzlich in einer Cloud neu zu entwickeln und bereitzustellen (Cloud-native); vgl. GONIWADA (2022), S. 17-26; HENNEBERGER (2016), S. 12-13.

Bei einer ausschließlichen Übertragung der bestehenden Anwendung in die Cloud können die technologischen Einschränkungen der monolithischen Softwarestruktur oftmals nicht oder nur partiell aufgelöst werden. Des Weiteren können technologische Weiterentwicklungen nur eingeschränkt genutzt werden, da sich Softwarebibliotheken, die in der Cloud zur Verfügung stehen, in einer monolithischen Anwendung nur mit einer großen Anpassung nutzen lassen. Auch die Vorteile der Skalierbarkeit können bei einer ausschließlichen Übertragung in die Cloud nur für die gesamte Anwendung und nicht für einzelne Funktionen vorgenommen werden. Für eine weitergehende Erläuterung der Vorteile von Cloud-nativen-Anwendungen gegenüber einem Cloud-enabling wird auf GONIWADA (2022), S. 19-20, verwiesen. Ein reines Cloud-enabling einer Anwendung schafft keinen Mehrwert hinsichtlich Innovationen, Bereitstellungsgeschwindigkeit und User-Experience; vgl. LÜNENDONK (2021), S. 14.

In den nachfolgenden Ausführungen wird ausschließlich auf die Vorgehensweise einer Neuimplementierung als Cloud-native-Anwendung eingegangen. Dies ist langfristig die tragfähigste Möglichkeit, eine Anwendung in die Cloud zu übertragen, um die technologischen Fortschritte, die sich durch die Cloud ergeben und zukünftig ergeben könnten, nutzen zu können; vgl. GONIWADA (2022), S. 17; HENNEBERGER (2016), S. 13.

Ziel ist es, den monolithischen Charakter des CBR-Tools jCORA zu überwinden. Die Funktionen von jCORA werden als Serverless-Funktionen in einer Cloud-Umgebung implementiert, sodass die Funktionen ausschließlich in der Cloud-Umgebung betrieben werden. Mittels eines webbasierten User Interfaces können die in der Cloud-Umgebung implementierten Funktionen

aufgerufen werden. Folgende zusätzliche Vorteile sollen neben der Auflösung der monolithischen Anwendungsstruktur durch die Konzipierung einer Cloud-nativen-Anwendung erreicht werden:

- Skalierbarkeit der Anwendung für den betrieblichen Anwendungszweck,
- Verbesserung der Wartbarkeit der Anwendung durch die klare Trennung zwischen Serverless-Funktionen und User Interface,
- Erhöhung der Nutzerakzeptanz durch ein anwenderfreundliches User Interface,
- Nutzung von frei verfügbaren KI-Entwicklungsbibliotheken sowie
- Kombination der Stärken von verschiedenen Cloud-Umgebungen.

Die nachfolgenden Ausführungen beschreiben konzeptionell, wie ein ontologiegestütztes Case-based Reasoning als Cloud-native-Anwendung implementiert werden kann. Ein vollumfänglicher Prototyp wird jedoch nicht erstellt. Stattdessen dienen die prototypisch entwickelten Funktionen sowie das als Klick-Prototyp konzipierte User Interface dazu, die Funktionsfähigkeit des ontologiegestützten Case-based Reasonings als Cloud-native-Anwendung in exemplarischer Weise zu demonstrieren (proof of concept). Des Weiteren sollen durch die Nutzung von weiteren KI-Entwicklungsbibliotheken die Vorteile einer Cloud-native-Anwendung aufgezeigt werden. Dazu wird exemplarisch eine spezifische Ähnlichkeitsfunktion implementiert, die mittels Künstlicher Neuronaler Netze die Ähnlichkeiten zwischen String-Werten für Instanzeigenschaften berechnet.

4.2 Cloud-Umgebungen

Aktuell wird der Cloud-Markt für frei zugängliche Cloud-Umgebungen im Wesentlichen durch drei Anbieter dominiert; vgl. SYNERGY RESEARCH GROUP (2022). Sie werden auch als Hyperscaler bezeichnet. Amazon Web Service (AWS) ist der Marktführer mit 34 % Marktanteil. Mit 21 % Marktanteil folgt Microsoft Azure. Auf dem dritten Rang befindet sich die Google Cloud Platform mit 10 % Marktanteil. Die drei Anbieter zusammen machen 65 % des weltweiten Cloud-Marktes aus. Darüber hinaus existieren weitere kleinere Anbieter, wie z. B. Alibaba Cloud und IBM Cloud, auf die nachfolgend jedoch nicht näher eingegangen wird.

Die Cloud-Umgebungen transformieren die aktuelle Informationstechnologie maßgeblich. Nach einer Studie von FORTUNE BUSINESS INSIGHTS (2023) betrug das Marktvolumen der Cloud-Technologie im Jahr 2022 weltweit 677,95 Milliarden US-$. Bis zum Jahr 2030 soll der Markt auf 2432,87 Milliarden US-$ anwachsen. Das Wachstum wurde zwar durch die Corona-Pandemie beschleunigt, begründet sich jedoch hauptsächlich durch die zunehmende Digitalisierung. Ein wichtiger Faktor für das potenzielle Wachstum wird in der Integration von Techniken der Artificial Intelligence (AI) in der Cloud gesehen.

Durch die zunehmende Nutzung der Cloud wächst auch das Datenvolumen in der Cloud stark an, das mittels Analysewerkzeugen aufbereitet werden muss. Vgl. PERTLWIESER (2022), S. 33. Die AI-Techniken unterstützen bei der Analyse der Daten. So entsteht ein starkes Zusammenwirken zwischen Cloud und AI-Techniken. Die Hyperscaler verfügen über verschiedene öffentlich nutzbare Bausteine von AI-Techniken, beispielweise in Form von Entwicklungsbibliotheken. Entsprechend wächst die Aufmerksamkeit für die sogenannten Cloud-basierten Developer Services zur Nutzung von AI-Techniken, die in verschiedenen Studien separat untersucht werden. Vgl. PERTLWIESER (2022), S. 33. Eine Studie, die speziell die Entwicklungsmöglichkeiten von AI-Techniken in der Cloud untersucht, ist beispielsweise der Special Report GARTNER (2022a) zu Cloud AI Developer Services.

Somit sind zwei Sichtweisen für die Betrachtung von Hyperscalern von Bedeutung: Neben der Nutzung einer Cloud-Umgebung sollten die Cloud-Umgebungen für Entwicklungen, insbesondere im Feld der AI-Techniken, zugänglich sein.

Amazon Web Service (AWS) war der erste Cloud Anbieter (Amazon Web Service existiert seit dem Jahr 2006) auf dem Markt und ist aktuell der Marktführer im Bereich Cloud; vgl. BÖGELSACK et al. (2022), S. 8-9. Amazon Web Service bietet seine Cloud auf global verteilten Rechenzentren an (derzeit 34 Standorte), die auf allen Kontinenten, ausgenommen der Antarktis, vertreten sind; vgl. AMAZON WEB SERVICES, INC. (2022e). Amazon Web Services ist eine Tochtergesellschaft von Amazon und die profitabelste Geschäftssparte des Unternehmens. Mittlerweile generiert die Cloud-Umgebung mehr als die Hälfte des operativen Gewinns des Unternehmens; vgl. AMAZON (2022), S .64-65.

Neben eigenen Diensten bietet Amazon Web Service (AWS) die Möglichkeit, von Kunden entwickelte Cloud-native-Anwendungen über die Cloud-Umgebung auf speziellen Marktplätzen anzubieten und so anderen Cloud-Nutzern zur Verfügung zu stellen. Zu den bekanntesten Kunden, die auf Basis der Amazon-Server ihre Dienste anbieten, gehören beispielsweise Netflix, Disney+, Delivery Hero, LinkedIn, Facebook und Twitter; vgl. AMAZON WEB SERVICES, INC. (2022c).

Die Google Cloud Platform (GCP) ist Teil der Google Cloud, die Google für die Erbringung seiner eigenen Dienste nutzt, wie z. B. für YouTube und Google Maps; vgl. BÖGELSACK et al. (2022), S. 13. Die Google Cloud Platform existiert seit dem Jahr 2008. Ähnlich wie Amazon bietet Google auf seiner Cloud-Umgebung den Zugriff auf verschiedene Softwareprodukte von Google; vgl. GOOGLE CLOUD (2022a). Für die Entwicklung einer Cloud-native-Anwendung mittels einer gängigen Programmiersprache ist vor allem die Entwicklungsumgebung Google Colab von Bedeutung. Google bietet seine Cloud-Umgebung in global verteilten Rechenzentren an (derzeit 34 Standorte), die – bis auf Afrika und Antarktis – auf allen Kontinenten vertreten sind; vgl. GOOGLE CLOUD (2022b). Google ist zwar erst nach Amazon (AWS) und Microsoft (Azure) in den Cloud-Markt eingestiegen, besitzt jedoch mit einer Wachstumsrate von 46 % im ersten Quartal 2021 das stärkste Wachstum gegenüber den anderen Cloud-Anbietern; vgl. REGENFUß/NINK (2022).

Die Google Cloud Platform zeichnet sich in den Bereichen von Big Data und Künstlicher Intelligenz als der Vorreiter aus. Dies beruht auf der Historie von Google im Bereich der Suchmaschinen; vgl. BÖGELSACK et al. (2022), S. 13; REGENFUß/NINK (2022). In diversen Anwendungsfällen wird die Google Cloud Platform in Multi-Cloud-Szenarien häufiger als sekundärer Anbieter oder ergänzend für spezialisierte Lösungen eingesetzt; vgl. REGENFUß/NINK (2022). Darüber hinaus besitzt Google im Gegensatz zu anderen Cloud-Anbietern ein eigenes hochverfügbares Leitungsnetz (einschließlich Seekabeln) mit Geschwindigkeiten von bis zu 250 TBit/s, das die Kontinente miteinander verbindet; vgl. GOOGLE CLOUD (2020). Das Google-eigene Leitungsnetz spielt für niedrige Latenzen und hohe Redundanz eine zentrale Rolle.

Microsoft startete seine Cloud unter dem aktuellen Namen „Microsoft Azure“ bereits im Jahr 2008. Ähnlich wie die zuvor genannten beiden Cloud-Anbieter stellt Microsoft mehrere Produkte von Microsoft in der Cloud zur Verfügung; vgl. MICROSOFT (2023b). Da eine Vielzahl von Unternehmen Microsoft-Produkte nutzt, liegt die Integration von Microsoft-Produkten mit Cloud-Diensten sehr nahe. Dies wird in mehreren Studien und Quellen auch als eine Stärke der Cloud von Microsoft gesehen; vgl. BÖGELSACK et al. (2022), S. 11; GARTNER (2022b). Anwender von Microsoft Azure können so die vorhandenen Lizenzen von Microsoft in der Azure Cloud weiter nutzen oder einen Rabatt erhalten, sollte eine spezielle Lizenzierung notwendig sein. Microsoft stellt die Microsoft Azure Cloud in verschiedenen Rechenzentren bereit. Die Standorte der Rechenzentren sind auf allen Kontinenten, bis auf die Antarktis, vertreten; vgl. MICROSOFT (2023a).

Grundsätzlich bieten die drei vorgenannten Cloud-Anbieter eine hohe Verfügbarkeit ihrer Cloud-Umgebungen sowie zahlreiche Produkte für die Nutzung der Cloud-Dienstleistungen an. Für die Konzipierung eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung wurden zwei Cloud-Anbieter ausgewählt, in deren Cloud-Umgebungen exemplarische Serverless-Funktionen für die Nutzung als Cloud-native-Anwendung entwickelt werden. Eine solche Multi-Cloud-Umgebung bietet Vorteile gegenüber einem einzigen Cloud-Anbieter. Dazu gehören insbesondere:

- die Nutzung von unterschiedlichen Stärken der unterschiedlichen Cloud-Anbieter,
- geringere Abhängigkeit von einem einzelnen Cloud-Anbieter sowie
- höhere Verfügbarkeit und Ausfallsicherheit durch Redundanzen.

Auf diese Weise wird exemplarisch aufgezeigt, wie sich die speziellen Stärken von einzelnen Cloud-Anbietern in einer Multi-Cloud-Umgebung nutzen lassen. Eine ähnliche Vorgehensweise findet sich auch in KUNSCHKE/SPITZ/POHLE (2022), S. 403-408.

Für die Konzipierung eines ontologiegestützten Case-based Reasonings als Cloud-native-Anwendung wird ein Cloud-Anbieter als primärer Anbieter ausgewählt. Der zweite Cloud-Anbieter dient als sekundärer Anbieter. Bei dem primären Anbieter wird die Mehrzahl der Serverless-Funktionen implementiert. Beim sekundären Anbieter werden spezielle Serverless-Funktionen implementiert, um einzelne Vorteile seiner Cloud-Plattform zu nutzen.

Für die Auswahl der beiden Cloud-Anbieter werden die nachfolgenden Kriterien zugrunde gelegt:

- Verfügbarkeit der Cloud-Umgebung,
- Nutzungskosten,
- intuitive Nutzbarkeit der Cloud,
- Marktanteil im Cloudmarkt,
- Zukunftssicherheit der Cloud,
- browserbasierte Entwicklungsumgebung,
- Innovationsoffenheit insbesondere für die Unterstützung von AI-Techniken sowie
- Möglichkeit, die Daten ausschließlich in deutschen oder europäischen Rechenzentren bereitzustellen.

Es wurden die Plattformen Amazon Web Service (AWS) und die Google Cloud Platform (GCP) ausgewählt. Der Amazon Web Service wird als primäre Plattform genutzt, in der die wesentlichen Funktionen (beispielsweise das Einlesen einer Ontologie, der Zugriff auf die Ontologie, der Ähnlichkeitsalgorithmus sowie spezifische Ähnlichkeitsfunktionen) implementiert werden. In der Google Cloud Platform werden ausschließlich spezifische Ähnlichkeitsfunktionen implementiert, die auf Basis von Künstlichen Neuronalen Netzen entwickelt werden. Die Google Cloud Platform wird für die Konzipierung des ontologiegestützten CBR-Systems als Cloud-native-Anwendung als sekundärer Anbieter ergänzend für spezialisierte Lösungen im Bereich der Künstlichen Neuronalen Netze eingesetzt. Die nachfolgende Abbildung 102 illustriert die Verwendung der beiden Cloud-Anbieter als Primär- und Sekundäranbieter.

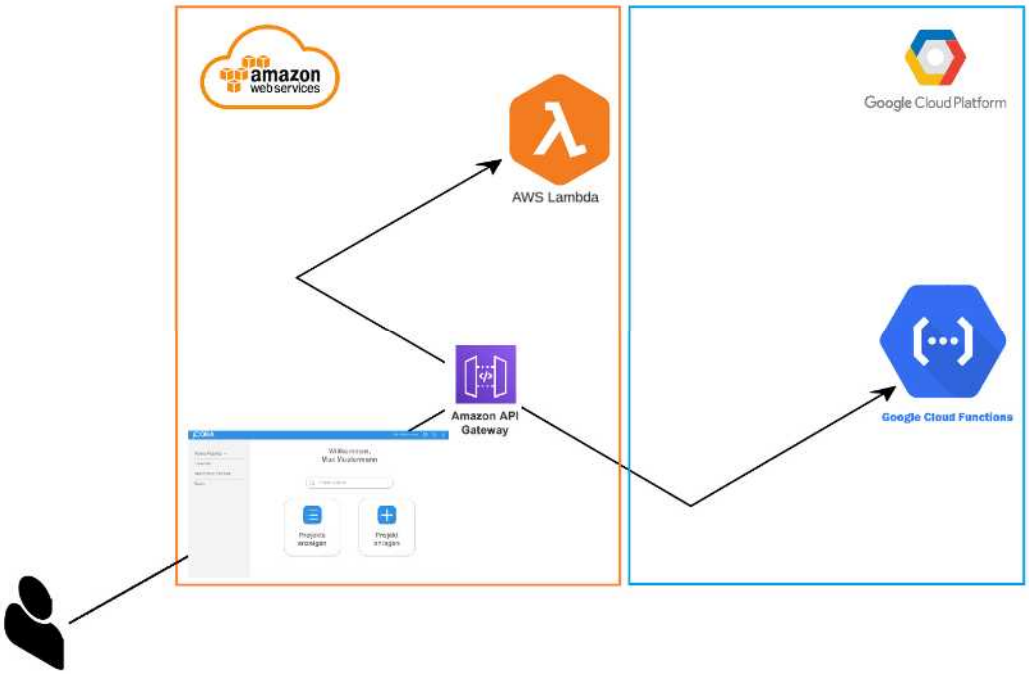


Abbildung 102: Konzipierte Multi-Cloud-Umgebung

Zunächst wird die Auswahl von Amazon Web Service (AWS) als primärer Anbieter erläutert:

- Amazon Web Service stellt den Marktführer mit einem Marktanteil von 34 % dar. Somit stellt Amazon Web Service über ein Drittel der genutzten Cloud-Umgebungen weltweit zur Verfügung mit einem Abstand von ca. 10 Prozentpunkten zu Microsoft Azure. Vgl. SYNERGY RESEARCH GROUP (2022).
- Amazon Web Service garantiert eine Verfügbarkeit von 99,9 %; vgl. AMAZON WEB SERVICES, INC. (2022b).
- Amazon Web Service ermöglicht die Auswahl von verschiedenen Standorten für die Bereitstellung der implementierten Serverless-Funktionen. Bei der Bereitstellung in einem deutschen oder europäischen AWS-Standort unterliegen die entwickelten Funktionen den deutschen Datenschutzbestimmungen.
- Die Preisgestaltung der verschiedenen Cloud-Anbieter wird in zahlreichen Publikationen kritisiert; vgl. GARTNER (2022b); LINTHICUM (2022); URBAN/GARLOFF (2022), S. 617; GLEB (2021), S. 53. Amazon Web Service bietet jedoch für geringe Lasten eine kostenfreie Nutzung an, die sich für eine Untersuchung konzeptioneller Art (wie im hier vorliegenden „proof of concept“) aufgrund der zu erwartenden geringen Lasten anbietet. Für die prototypische Implementierung der einzelnen Serverless-Funktionen in der AWS-Cloud entstehen dadurch keine Kosten.

- Die Entwicklungsumgebung Cloud9 von Amazon Web Service ist eine vollintegrierte, browserbasierte Entwicklungsumgebung in der AWS-Cloud, die es ermöglicht, mittels gängiger Programmiersprachen Anwendungen zu entwickeln und als Serverless-Funktionen zu implementieren.
- Die Nutzung der AWS-Cloud wird in der Fachliteratur oftmals als intuitiv gewürdigt; vgl. BÖGELSACK et al. (2022), S. 123; POTHECARY (2021), S. 111. Darüber hinaus existieren zahlreiche Dokumentationen sowie eine breite Einführungsliteratur für die Nutzung von Amazon Web Service; vgl. beispielsweise AMAZON WEB SERVICES, INC. (2022d).

Insgesamt betrachtet, ist Amazon Web Service als marktführender Cloud-Anbieter insbesondere wegen der Preisgestaltung für prototypische Implementierungen und wegen der vollintegrierten Entwicklungsumgebung als primärer Cloud-Anbieter für die Entwicklung einer Cloud-native-Anwendung am besten geeignet.

Die Google Cloud Platform wird als sekundärer Cloud-Anbieter ausgewählt, um die zuvor genannten Vorteile einer Multi-Cloud-Umgebung zu nutzen. Insbesondere die Verwendung von Entwicklungsbibliotheken für AI-Techniken und die umfangreiche Dokumentation mittels der browserbasierten Entwicklungsumgebung von Google Cloud Platform stellen eine wesentliche Stärke dieses Cloud-Anbieters dar. Die Google Cloud Platform wird auch von GARTNER (2022a) als führende Cloud-Umgebung im Bereich von AI-Techniken angesehen.

Die frei verfügbaren Code-Beispiele auf der Google Cloud Platform machen es den Entwicklern einfach, große Datenmengen in Echtzeit aus unterschiedlichen Quellen zu nutzen und die Nutzung der AI-Entwicklungsbibliotheken zu verstehen. Die AI-Entwicklungsbibliotheken wurden von Google selbst erstellt, sind für die Verwendung als Entwicklungsbibliotheken in der „modernen“ Programmiersprache Python (hierauf wird später zurückgekommen) vorgesehen und stehen für die öffentliche Nutzung frei zur Verfügung. Insbesondere die AI-Entwicklungsbibliothek „Word2Vec“ besitzt für die vorliegende Untersuchung eine herausragende Bedeutung. Darauf wird später ausführlich eingegangen. Insgesamt betrachtet, wird der spezielle Vorteil der Google Cloud Platform hinsichtlich der gut dokumentierten und mit Code-Beispielen illustrierten AI-Techniken gesehen.

Zusammenfassend ist festzuhalten, dass Amazon Web Service als etablierter Anbieter und wegen der zuvor genannten Vorteile für die Wahl des primären Cloud-Anbieters für die Konzipierung eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung zu empfehlen ist. Für die Implementierung von einzelnen Serverless-Funktionen, die auf TensorFlow und Word2Vec basieren (Erläuterungen folgen später), eignet sich dagegen die Google Cloud Platform als sekundärer Cloud-Anbieter. Dies begründet die Konzipierung eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung mittels einer Multi-Cloud-Umgebung.

4.3 Konzipierung eines ontologiegestützten Case-based-Reasoning-Systems als Cloud-native-Anwendung

4.3.1 Konzipierung des Frontends

Es existiert bereits eine Konzeption für das Frontend – das User Interface (UI) – für ein ontologiegestütztes CBR-System als Cloud-native-Anwendung. Diese Konzeption wurde in WEBER et al. (2023), S. 37-102, als ein „Klick-Prototyp“ vorgestellt. Für die Prototyp-Entwicklung wurde der Usability-Engineering-Prozess nach NIELSEN zugrunde gelegt. Im Folgenden werden die wichtigsten Ergebnisse aus der vorgenannten Publikation kurz erläutert.

Im ersten Schritt wurden die Heuristiken in Anlehnung an NIELSEN definiert. NIELSEN empfiehlt zehn Heuristiken zur Durchführung einer heuristischen Evaluation, um Usability-Probleme einer Anwendung identifizieren zu können. Diese Heuristiken decken Problemkategorien ab, die bei der Gestaltung einer Anwendung berücksichtigt werden sollten. Die zehn Heuristiken von NIELSEN werden um zwei weitere Heuristiken ergänzt. Diese insgesamt zwölf Heuristiken bilden die Grundlage für die Usability-Evaluation des bislang erörterten, prototypischen CBR-Tools jCORA, das als Ausgangspunkt für die Konzipierung von Frontend und Backend eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung dient. Die Usability-Evaluation wird von Experten durchgeführt, die sich in die Rolle eines (End-)Nutzers versetzen. Die Experten untersuchen ein System auf Verletzungen der Heuristiken. Die zwölf Heuristiken spiegeln die gewünschten Eigenschaften der Interaktion zwischen einem (End-)Nutzer und einem System wider. Wird eine Verletzung einer dieser gewünschten Eigenschaften identifiziert, ist dies ein Hinweis auf ein mögliches Usability-Problem. Ziel der heuristischen Evaluation ist es, alle Usability-Probleme zu identifizieren. Vorrangig werden solche Usability-Probleme berücksichtigt, die einen einschränkenden Einfluss auf die Gebrauchstauglichkeit eines Systems haben. Dieses System stellt hier das CBR-Tool jCORA dar. Die Ergebnisse der heuristischen Evaluation werden in der nachfolgenden Tabelle 56 dargestellt:

	Heuristik	Usability-Problem (ja/nein)	Problembenennung
Heuristiken nach NIELSEN	1. **Sichtbarkeit des Systemstatus**	nein	
	2. **Übereinstimmung zwischen System und realer Welt**	ja	Die Begriffe „Konzept“, „Relation“ „nicht taxonomische Relation“, „Instanz“ und „Attribut“ stellen keine Begriffe dar, die einem Projektmanager vertraut sind.

	3. **Benutzerkontrolle und Freiheit**	nein	
	4. **Konsistenz und Standards**	ja	Das Hinzufügen von „Relationen“, „Instanzen“ und „Attributen“ erfolgt auf eine unterschiedliche Art und Weise.
	5. **Fehler vermeiden**	nein	
	6. **Wiedererkennen statt Erinnern oder Erkennen vor Erinnern**	ja	Aktionen sind in jCORA 1.2.5 nicht leicht auffindbar.
	7. **Flexibilität und effiziente Nutzung**	nein	
	8. **Ästhetik und minimalistisches Design**	ja	Nicht alle zur Nutzung relevanten Informationen sind vorhanden.
	9. **Hilfe bei der Fehlerbehebung**	ja	Fehlermeldungen erweisen sich des Öfteren als unverständlich.
	10. **Hilfe und Dokumentation**	ja	Der Hilfe-Button weist keine Funktion auf.
Weitere Heuristiken	11. **Wahrnehmungssteuerung**	nein	
	12. **Joy of Use**	ja	Die Gestaltung des CBR-Tools jCORA mutet nicht „zeitgemäß“ an.

Tabelle 56: Heuristiken zur Problemidentifikation für jCORA

Im zweiten Schritt wurde ein Usability-Test in Form eines Feldtests durchgeführt. Dazu wurde ein Umfragetool erstellt und von den Testpersonen beantwortet. Die Ergebnisse des Usability-Tests zeigen, dass Usability-Probleme bei wesentlichen Funktionen des CBR-Tools jCORA bestehen, insbesondere hinsichtlich der Problembeschreibung und der Ähnlichkeitsberechnung.

Insgesamt lag die Fehlerquote bei 47 %. Das Ergebnis des Feldtests wird in der nachfolgenden Tabelle 57 dargestellt.

Usability-Komponente	**Indikatoren**	**Ergebnis des Usability-Tests zu jCORA** (Durchschnittswerte)
Effektivität	Vollständigkeit der Aufgabenbearbeitung	92 %
	Fehlerquote	47 %
Effizienz	Zeit der Bearbeitung der Aufgaben	9 min
Zufriedenstellung	Wiederverwendung	80 %
	Weiterempfehlen an Kollegen	60 %
	Zufriedenheit	40 %

Tabelle 57: Ergebnisse des Feldtests für jCORA

Eine weitere Erkenntnis des Feldtests, die nicht direkt auf die Usability-Problematik des CBR-Tools jCORA abzielt, ist es, dass ein grundsätzlicher Bedarf an einem CBR-Tool wie jCORA besteht, auch wenn dieses CBR-Tool hinsichtlich seiner Usability als unzureichend bewertet wird.

Der dritte Schritt erstreckte sich auf die Analyse der Usability-Probleme des CBR-Tools jCORA. Dazu wurden Usability-Ziele definiert, die im Hinblick auf eine Usability-Verbesserung durch die Lösung der bereits skizzierten Usability-Probleme von jCORA erreicht werden sollten. Dies bedeutet, dass der Usability-Test für den Klick-Prototyp in Bezug auf Effektivität, Effizienz und Zufriedenheit mindestens die gleichen Werte erzielen sollte wie der Usability-Test für jCORA. Ziel war es, eine deutlich höhere Usability für den Klick-Prototyp zu erreichen. Dazu sollten die Werte in der Tabelle57 übertroffen werden. Des Weiteren wurden im Rahmen der Analyse fünf User Stories formuliert, die eine Grundlage für das Design eines Klick-Prototyps darstellen sollten, da in diesen User Stories Funktionalitäten formuliert wurden, die jCORA derzeit nicht besitzt und die später im Klick-Prototyp ergänzt werden sollen. Die nachfolgende Tabelle 58 zeigt die User Stories, die der Analyse der Usability-Probleme des CBR-Tools jCORA zugrunde liegen.

(End-)Nutzerrolle	Ziel	Grund
Senior Sales Manager	• Status eines Projektes • meine Projekte	• schnellere Erfassung der Bedeutung eines Projekts • zügigere Anpassung eigener Projekte
Solution Manager	• qualifiziertes Auffinden vergleichbare Inhalte • Erfassung eigener Bewertungen	• Zeitersparnis • Standardisierung • Qualitätskontrolle
Technical Consultant	mein Know-how rund um intuitives Handling und zwischen-menschliches Verstehen	Kunden ihre Arbeit erleichtern
Client & Bid Manager	Referenz-Vergleiche	für jede weitere Ausschreibung ganz einfach die passenden Referenzen finden, welche bereits aufbereitet wurden.
Teilprojektleiter & Business Analyst	Suchen nach Lösungswegen, welche die Herausforderungen im Datenkontext (Datenmigration, Datenpflege, Datenkonvertierung) gelöst haben	Identifikation von Potenzial für Lösungen bei bereits umgesetzten Projekten

Tabelle 58: User Stories zur Analyse der Usability-Probleme von jCORA

Die Usability-Ziele sowie die User Stories aus dem Analyseschritt flossen in den vierten Schritt ein, in dem mehrere Designvorschläge entwickelt wurden, die sich an gängigen Wissensmanagement-Tools orientieren. Weitere Designentscheidungen betrafen z. B. Typographie und Symbole.

Unter Berücksichtigung der User Stories, der Usability-Ziele und des Designvorschlags wurde mithilfe der Software Adobe-XD ein Klick-Prototyp mit insgesamt 532 Slides entwickelt. Die nachfolgende Abbildung 103 zeigt exemplarisch ein Slide des Klick-Prototyps anhand der projektbezogenen Fallspezifizierung; vgl. WEBER et al. (2023), S. 81.

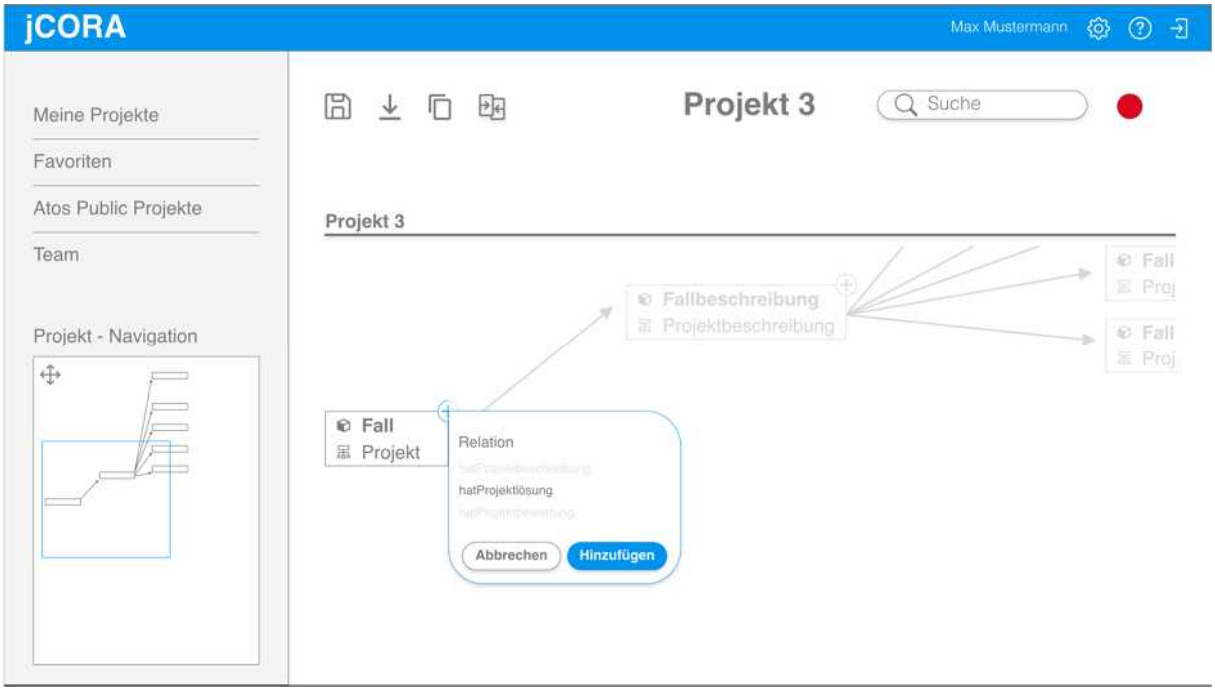


Abbildung 103: Klick-Prototyp

Der Klick-Prototyp wurde – analog zum vorherigen Vorgehen hinsichtlich des CBR-Tools jCORA – zunächst einer heuristischen Evaluation und anschließend einem Usability-Test unterzogen. Die heuristische Evaluation zeigte, dass für den neuen Klick-Prototyp nur noch eine Usability-Fehlerkategorie existiert, nämlich die oftmals nicht bekannten, fachsprachlichen Bezeichnungen von Ontologiekomponenten, wie vor allem „Klasse“, „nicht taxonomische Relation“, „Instanz“ und „Attribut“; siehe hierzu die Heuristik „Übereinstimmung zwischen System und realer Welt“ in der nachfolgenden Tabelle 59.

Die Ergebnisse der heuristischen Expertenevaluationen werden in der nachfolgenden Tabelle 59 zusammengefasst. In dieser Tabelle werden die Ergebnisse für das prototypische CBR-Tool jCORA einerseits und den neu designten Klick-Prototyp andererseits einander gegenübergestellt.

	Heuristik	Usability-Problem (ja/nein) für das Tool **jCORA**	Usability-Problem (ja/nein) für den **Klick-Prototyp**
Heuristiken nach NIELSEN	1. Sichtbarkeit des Systemstatus	nein	nein
	2. Übereinstimmung zwischen System und realer Welt	ja	ja
	3. Benutzerkontrolle und Freiheit	nein	nein
	4. Konsistenz und Standards	ja	nein
	5. Fehler vermeiden	nein	nein
	6. Wiedererkennen statt Erinnern oder Erkennen vor Erinnern	ja	nein
	7. Flexibilität und effiziente Nutzung:	nein	nein
	8. Ästhetik und minimalistisches Design	ja	nein
	9. Hilfe bei der Fehlerbehebung	ja	nein
	10. Hilfe und Dokumentation	ja	nein
weitere Heuristiken	11. Wahrnehmungssteuerung	nein	nein
	12. Joy of Use	ja	nein

Tabelle 59: Vergleich der Usability-Probleme
zwischen dem CBR-Tool jCORA und dem Klick-Prototyp

Bei der Auswertung des Usability-Tests für den neuen Klick-Prototyp zeigte sich eine deutliche Verbesserung. Die Fehlerquote lag nur noch bei 8 % (beim ersten Feldtest mit jCORA lag dieser bei 47 %), die Weiterempfehlungsrate der Anwendung stieg ebenfalls an, was den grundsätzlichen Bedarf einer solchen Anwendung weiter unterstreicht. Die nachfolgende Tabelle 60 zeigt den Vergleich des Feldtests zwischen dem CBR-Tool jCORA und dem Klick-Prototyp als überarbeitetem User Interface. Verbesserungen der Indikatoren für Usability-Komponenten sind in der Spalte des Klick-Prototyps jeweils fett hervorgehoben.

Usability-Komponente	**Indikatoren**	**Ergebnis des Usability-Tests zu jCORA** (Durchschnittswerte)	**Ergebnis des Usability-Tests zum Klick-Prototyp** (Durchschnittswerte)
Effektivität	Vollständigkeit der Aufgabenbearbeitung	92 %	**96 %**
	Fehlerquote	47 %	**8 %**
Effizienz	Zeit der Bearbeitung der Aufgaben	9 min	**7 min**
Zufriedenstellung	Wiederverwendung	80 %	80 %
	Weiterempfehlen an Kollegen	60 %	**80 %**
	Zufriedenheit	40 %	**80 %**

Tabelle 60: Vergleich der Feldtests
zwischen dem CBR-Tool jCORA und dem Klick-Prototyp

Mit dem Klick-Prototyp konnte eine deutliche Usability-Verbesserung gegenüber dem prototypischen CBR-Tool jCORA erreicht werden. Die hiermit gesammelten Erkenntnisse stellen eine Grundlage dar, die sich zur benutzerfreundlichen Gestaltung des Frontends für ein zukünftiges, professionell implementiertes ontologiegestütztes CBR-System als Cloud-native-Anwendung nutzen lässt.

Einschränkend muss jedoch auch erwähnt werden, dass der Klick-Prototyp zwar eine mögliche Gestaltung des User Interfaces demonstriert, jedoch nicht abschließend das Frontend einer Anwendung darstellt. Ein Klick-Prototyp ist zwar hilfreich, um das User Interface und die Benutzerinteraktionen zu demonstrieren, kann jedoch nicht abschließend das Frontend einer Anwendung repräsentieren. Dies liegt daran, dass Klick-Prototypen in der Regel auf die Darstellung von User Interface und Interaktionen beschränkt sind, aber nicht die konkrete Funktionalität abdecken, die für die Integration eines Backends erforderlich ist. Darüber hinaus bieten Klick-Prototypen nur eine statische Darstellung der Benutzeroberfläche, während „echte“ Anwendungen dynamische Elemente, Echtzeitdatenverarbeitung, Sicherheitsaspekte und Leistungsoptimierungen erfordern, die erst während der Implementierung auftreten und möglicherweise Anpassungen der Benutzeroberfläche erfordern. Aus den vorgenannten Gründen können sich bei der realen Implementierung eines Frontends Probleme ergeben, die eine Anpassung des User Interface erfordern und widrigenfalls zu Lasten der oben betrachteten Heuristiken gehen. Auf-

grund dieser fehlenden realen Implementierung kann deshalb letztlich nicht sichergestellt werden, dass sich das als Klick-Prototyp zur Verfügung gestellte User Interface vollständig in ein ontologiegestütztes CBR-System als Cloud-native-Anwendung integrieren lässt.

4.3.2 Konzipierung des Backends

4.3.2.1 Vorüberlegungen zur Backend-Konzipierung

Die Clients des Frontends eines ontologiegestützten CBR-Systems greifen über Methoden jeweils eines Application Programming Interfaces (API) auf ein Backend zu. Ein Client kann beispielsweise eine für mobile Geräte ausgelegte Anwendung, eine ausschließlich für das Web ausgelegte Anwendung oder ein spezifischer Client innerhalb einer bestehenden unternehmensbezogenen Anwendung, wie z. B. SAP oder Sharepoint, sein. In den nachfolgenden Ausführungen werden die API-Methoden erläutert, die durch ein Representational State Transfer Application Programming Interface (RESTful-API) bereitgestellt werden. Die RESTful-API ist eine Schnittstelle zwischen IT-Systemen (hier zwischen Client und Backend), die das Hypertext Transfer Protocol (HTTP) für ihre Kommunikation verwendet und Daten mithilfe von JSON (JavaScript Object Notation) überträgt.

Ein Backend lässt sich in eine Middleware- und eine Datenbank-Komponente unterteilen.

Die Datenbank-Komponente des Backends ist in der nachfolgenden Betrachtung unerheblich, da die Datenbank ausschließlich für die Speicherung der Daten vorgesehen ist und hier keinerlei weitere Betrachtung hinsichtlich der Datenspeicherung erfolgt. Die Bereitstellung einer standardisierten und kostenfreien Datenbank kann beispielsweise mittels eines Standardprodukts wie DynamoDB in der Cloud erfolgen. Die sicherheitskritische IT-Projekt-Ontologie lässt sich in dieser Datenbank vorhalten.

Die Middleware-Komponente des Backends setzt sich aus einem API-Gateway und mindestens einer Serverless-Funktion zusammen. Das API-Gateway dient als „Eingangstor“ aller Kommunikationsanfragen der Clients an das Backend. Das API-Gateway verarbeitet die Kommunikationsanfragen, die in einem vorgegebenen API-Format erwartet und durch das Hypertext Transfer Protocol (HTTP) transportiert werden. Die Anfragen werden durch das API-Gateway an die dahinter liegenden Serverless-Funktionen weitergeleitet. In der nachfolgenden Abbildung104 ist exemplarisch die Serverless-Funktion `ermittleKonzeptAehnlichkeit (`Serverless-Funktionen werden hier durch diese spezielle Schriftart gekennzeichnet`)` dargestellt. Ebenso wird der zugehörige Ressourcen-Pfad `/Konzeptaehnlichkeit (`Ressourcen und Ressourcenpfade werden durch diese spezielle Schriftart gekennzeichnet) angegeben`,` der zum Aufruf der Serverless-Funktion führt. Die Rückmeldungen der Serverless-Funktion stellen die Berechnungsergebnisse dar, die über das API-Gateway an die angefragten Clients übermittelt werden. Die nachfolgende Abbildung104 illustriert die zuvor erläuterte Funktionsweise der Middleware-Komponente des Backends eines ontologiegestützten CBR-Systems.

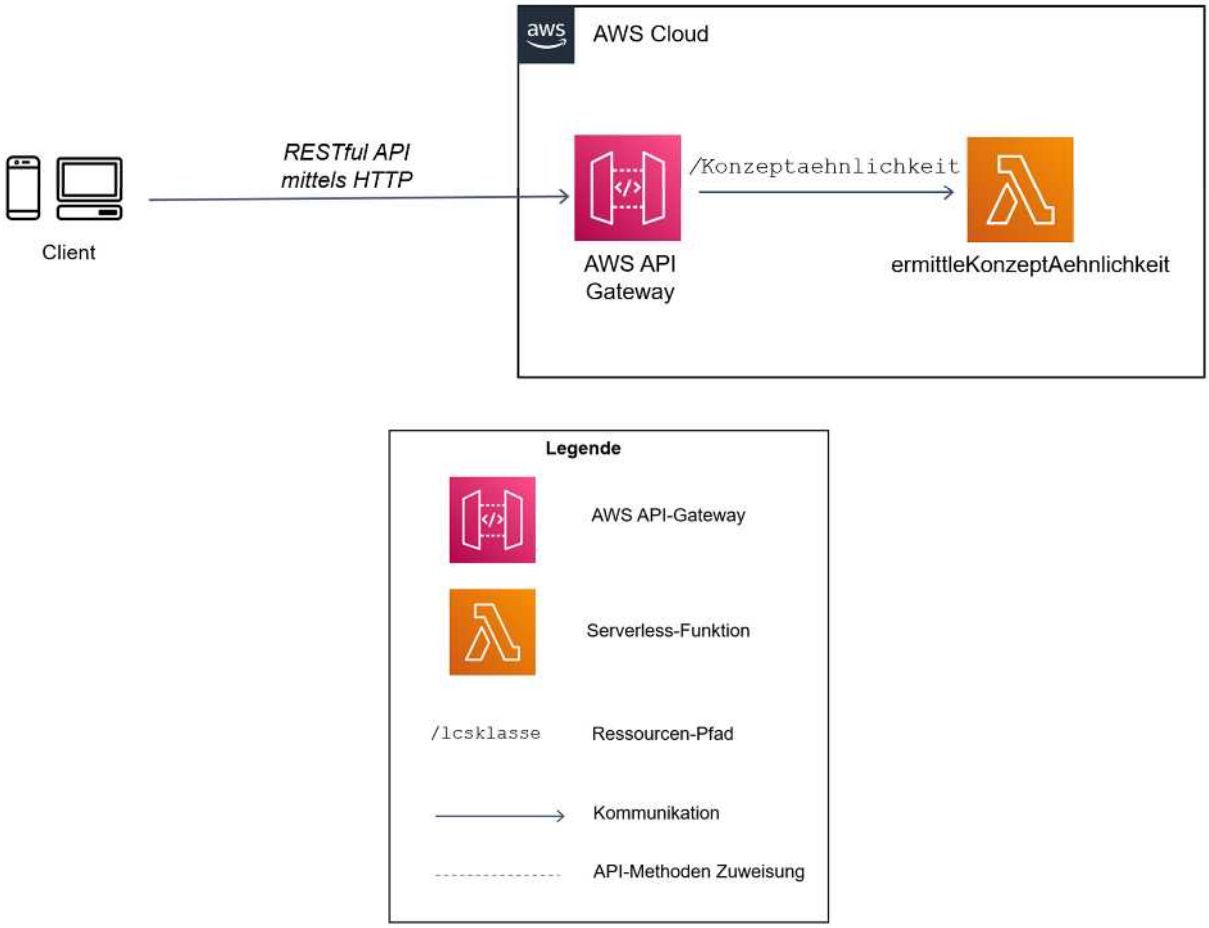


Abbildung 104: Darstellung der Middleware-Komponente des Backends eines ontologiegestützten CBR-Systems

Der Schwerpunkt der nachfolgenden Ausführungen liegt ausschließlich auf den Serverless-Funktionen, die in einer Cloud-Umgebung bereitgestellt werden und die Geschäftslogik eines Unternehmens abbilden, das sein Projektmanagement AI-basiert mithilfe von Ontologien und Case-based Reasoning durchführt.

Ein API-Gateway ist ein Teil eines API-Management-Tools, das zwischen einem Client und mehreren Serverless-Funktionen vermittelt. Es dient als zentrale Schnittstelle, die alle API-Methoden – wie PUT, ANY, GET, POST, PATCH, OPTIONS, HEAD und DELETE – von den Clients entgegennimmt, an die erforderlichen Serverless-Funktionen weiterleitet und die von diesen Funktionen berechneten Ergebnisse an die Clients zurückgibt. Ein API-Gateway betreibt eine Reihe von APIs an einem konkreten Knotenpunkt, der mittels eines Uniform Resource Locator (URL) abgerufen werden kann.

In der nachfolgenden Abbildung105 wird der Knotenpunkt „www.jcora.de“ exemplarisch dargestellt. In dieser Abbildung werden die API-Methoden GET, POST, ANY und DELETE beispielhaft angeführt. Die API-Methode GET wird mit der Serverless-Funktion `ermittleLCS-Klasse` verbunden, sodass bei einem Aufruf der GET-Methode (mittels des Ressourcen-Pfads

`www.jcora.de/lcsklasse`) die Serverless-Funktion **`ermittleLCSKlasse`** ausgeführt wird.

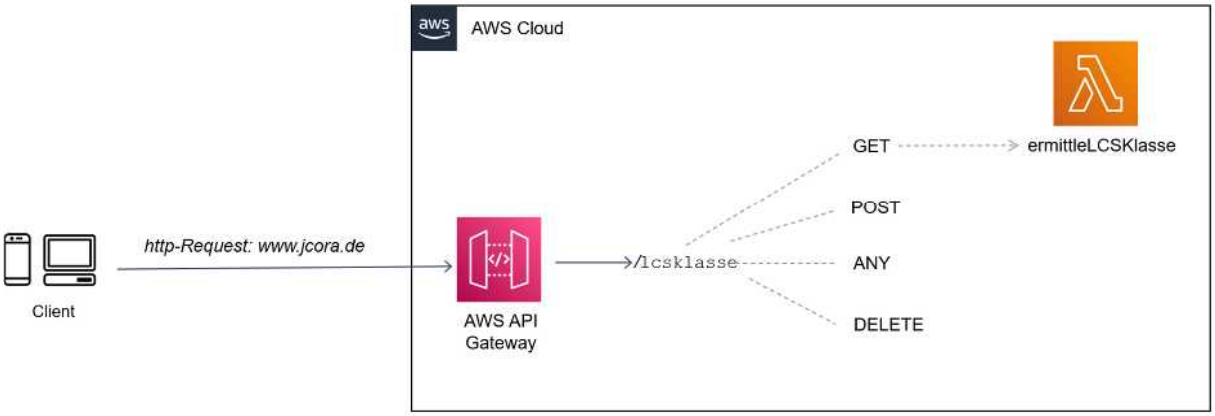


Abbildung 105: Zugriff auf die API der Ressource `lcsklasse`

Ein API-Gateway unterstützt zwei Typen von API-Methoden, und zwar RESTful- und WebSocket-API-Methoden. In diesem Beitrag sind nur die RESTful-API-Methoden von Bedeutung. Daher werden nachfolgend unter API-Methoden stets RESTful-API-Methoden verstanden.

Die nachfolgende Abbildung 106 zeigt exemplarisch die Konfiguration des API-Gateways für die Ressource `lcsklasse` mit dem Ressourcen-Pfad `/lcsklasse`. Für diese Ressource können verschiedene API-Methoden mit entsprechenden Serverless-Funktionen spezifiziert werden. In der Abbildung106 ist die GET-Methode mit der Serverless-Funktion **`ermittle LCSKlasse`** konfiguriert worden. Jede Ressource kann zwar mehrere API-Methoden besitzen, jedoch nur immer von einem Methodentyp. Daher kann die Ressource `lcsklasse` keine weitere GET-Methode besitzen.

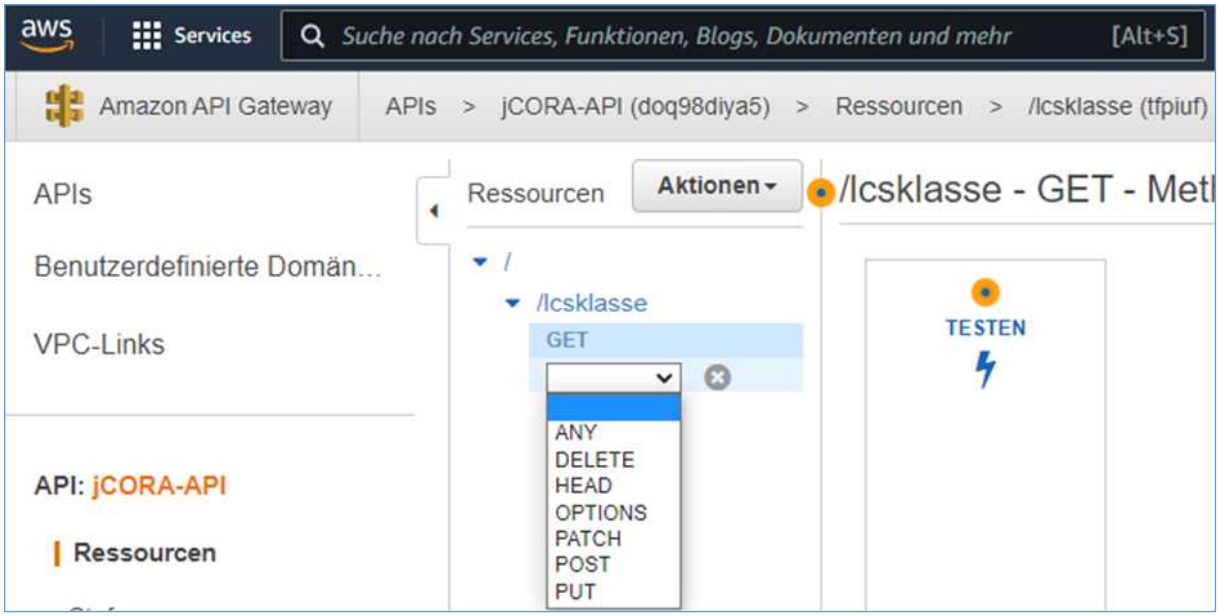


Abbildung 106: AWS-API-Gateway-Konfiguration

Die Hauptaufgabe des API-Gateways ist die Kapselung der Serverless-Funktionen für die Clients. Die Vorteile der Nutzung eines API-Gateways werden nachfolgend angeführt.

- Sicherheit: Durch die Verwendung eines API-Gateways müssen nicht alle Funktionalitäten der „Außenwelt“ zur Verfügung gestellt werden. Serverless-Funktionen sind für Clients nicht direkt zugänglich, sondern können nur über das API-Gateway mit vorheriger Autorisierung erreicht werden.
- Netzwerk-Routing: Bei einer direkten Kommunikation zwischen Client und Serverless-Funktionen erfordert eine Transaktion mehrere Funktionsaufrufe. Dieser Ansatz kann zu mehreren Netzwerk-Roundtrips zwischen dem Client und dem Server führen, was eine erheblich höhere Latenz nach sich zieht.
- Logging und Monitoring: Das API-Gateway kann als zentrales Logging und für das Überwachen des Einstiegspunktes dienen, um kritische Anwendungen durch einen zentralen Einstiegspunkt zu überwachen.
- Unabhängigkeit der Serverless-Funktionen von Endgeräten: Da der Zugriff auf die Serverless-Funktionen über das API-Gateway erfolgt, besteht keine harte Kopplung zwischen Frontend und Backend. Dies ermöglicht eine flexible Ausgestaltung der Clients.
- Weiterentwicklung einer Anwendung: Wenn Serverless-Funktionen weiterentwickelt werden, kann es bei einem direkten Aufruf aus einem Client zu Fehlerzuständen bei den Client-Anwendungen kommen, weil eine direkte Verknüpfung zwischen Client und Serverless-Funktion besteht. Mittels API-Gateways kann jedoch durch eine gleichbleibende API-Aufrufstruktur eine Anpassung der Serverless-Funktion geschehen, ohne einen Fehlerzustand zu erreichen, weil zwischen Client und Serverless-Funktion definiert ist, was übergeben und zurückgeliefert werden soll.

Die Nutzung einer RESTful-API-Methode für ein ontologiegestütztes CBR-System als Cloud-native-Anwendung erfordert zunächst die Spezifizierung des generellen Aufbaus eines HTTP-Requests und einer HTTP-Response.

- Ein HTTP-Request besteht aus einer „Request-Line“ (die den Aufruf der URL darstellt) und den „HTTP-Header-Feldern“. Optional kann ein „Message-Body“ vorhanden sein.
- Eine HTTP-Response besteht aus einer „Status-Line“, den „HTTP-Header-Feldern“ und einem „Content-Type“.

Die nachfolgende Abbildung 107 illustriert die zuvor erläuterten Sachverhalte.

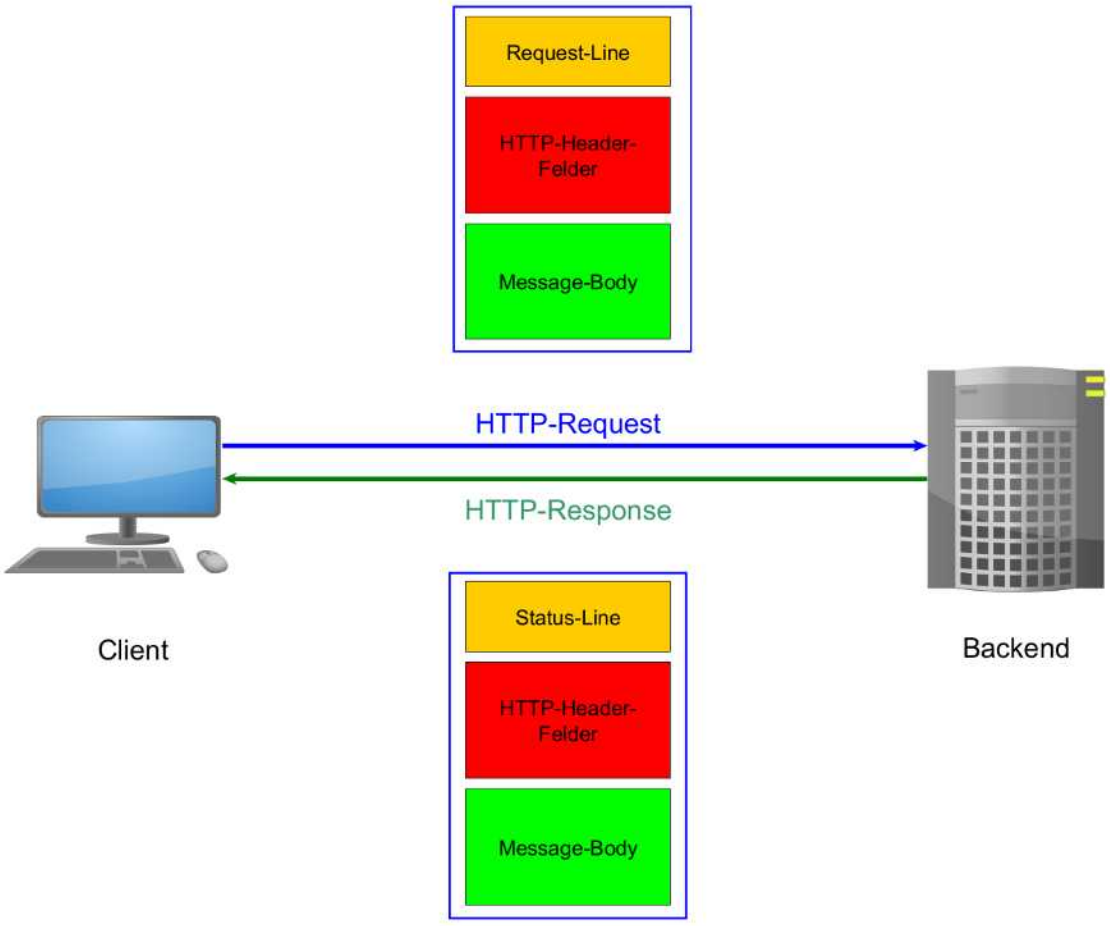


Abbildung 107: Client-Backend-Kommunikation mittels HTTP-Request und HTTP-Response

In den nachfolgenden Ausführungen wird auf die Konfiguration des API-Gateways, die RESTful-APIs mit den dazugehörigen HTTP-Requests und HTTP-Responses sowie die Serialisierung der Berechnungsergebnisse der Serverless-Funktionen in das JSON-Format für die Übertragung zwischen Backend und Frontend nicht weiter eingegangen, weil diese Aspekte für die Konzipierung eines Backends für ein ontologiegestütztes CBR-System als Cloud-native-Anwendung zwar eine nicht zu unterschätzende Bedeutung besitzen, aber für die nachfolgenden Untersuchungen keinen Schwerpunkt bilden. Dies wird wie folgt begründet:

- Die Geschäftslogik bilden die Serverless-Funktionen ab.

- Die Vorteile einer Cloud-Umgebung sind in den Serverless-Funktionen verankert.
- Die notwendigen Übergabeparameter im HTTP-Header lassen sich anhand der Übergabeparameter der Serverless-Funktionen ableiten.
- Die notwendigen Rückgabeparameter im HTTP-Response werden ebenfalls durch die Serverless-Funktionen bestimmt.
- Das Serialisieren der Objekte in ein JSON-Format lässt sich mittels einer Standardfunktion durchführen.
- Amazon Web Service bietet für die Nutzung des Amazon-eigenen API-Gateways eine „konfigurative" Vorgehensweise an, um erwartete Funktionsanforderungen (beispielsweise erwartete Übergabeparameter) und erwartete Funktionsantworten für eine implementierte Serverless-Funktion zu hinterlegen.

Die Herausforderung bei der Konzeption eines Backends für ein ontologiegestütztes CBR-System als Cloud-native-Anwendung liegt im Wesentlichen in der Entwicklung der Serverless-Funktionen zur Abbildung der Geschäftslogik. Die folgenden Ausführungen beziehen sich daher ausschließlich auf die Entwicklung solcher Serverless-Funktionen.

4.3.2.2 Konzipierung der Middleware für das Backend einer Cloud-native-Anwendung

Für die Implementierung der Serverless-Funktionen wird die Programmiersprache Python eingesetzt. Python stellt eine „moderne" objektorientierte Programmiersprache dar; vgl. DOWNEY (2021); KLEIN (2021); LUTZ (2007).

Die Auswahl der Programmiersprache Python für die Implementierung der Serverless-Funktionen eines ontologiegestützten CBR-Systems wird wie folgt begründet:

- Die Programmiersprache Python ist aktuell die meistgenutzte Programmiersprache für die Entwicklung von Backends; gl. CASS (2022); STACK OVERFLOW (2021). Gemäß der Studie von POPULARITY OF PROGRAMMING LANGUAGE (2023), Stand Oktober 2022, stellt Python die beliebteste Programmiersprache für Entwickler dar.
- Die Weiterentwicklung von Python wird von einer großen und aktiven Community angetrieben. Gemäß der Studie von SLASHDATA (2022), S. 13, Stand erstes Quartal 2022, hat Python die größte Backend-Entwickler-Community und die zweitgrößte Entwickler-Community insgesamt im Hinblick auf gängige Programmiersprachen.
- Die Python-Programmiersprache wird unter einer Open-Source-Lizenz vertrieben und kann ohne Lizenzkosten genutzt werden. Die geistigen Eigentumsrechte hinter der Programmiersprache Python besitzt die gemeinnützige Organisation „Python Software Foundation"; vgl. PYTHON (2023a). Sie verwaltet die Open-Source-Lizenzierung für die Python-Version 2.1 und höher; vgl. PYTHON (2023a). Im „Mission Statement" der Or-

ganisation ist formuliert, dass die zentrale Python-Distribution der gesamten Öffentlichkeit kostenlos zur Verfügung gestellt wird; vgl. PYTHON (2002). Dazu gehören die Python-Programmiersprache selbst, ihre Standardbibliotheken und -dokumentation, Installationsprogramme, Quellcode und Schulungsmaterialien; vgl. PYTHON (2002).

- Python wird von allen Cloud-Umgebungen voll unterstützt, jegliche Beispiele und Erläuterungen werden in Python dargestellt. Dadurch ist eine hohe Portabilität gegeben, weil eine Anwendung in verschiedenen Cloud-Umgebungen bereitgestellt werden kann. Beispielhaft wird auf die Entwicklungsbeispiele von Amazon Web Service und von Google Cloud Platform verwiesen; vgl. AMAZON WEB SERVICES, INC. (2022a) bzw. GOOGLE DEVELOPERS (2022).
- Python gilt als Programmiersprache, welche einen knappen und gut verständlichen Programmierstil fördert. Dadurch sind Python-basierte Anwendungen deutlich konziser formuliert als in anderen Programmiersprachen; vgl. PYTHON (2023b); STEYER (2018), S. 3.
- Python wird als gängige Programmiersprache für Cloud-basierte Entwicklungen insbesondere im Umfeld von Künstlicher Intelligenz und für Datenanalysen angesehen; vgl. SLASHDATA (2022), S. 13. Diverse praxisorientierte Einführungsliteratur zu KI-Techniken – wie FROCHTE (2021); RASCHKA/MIRJALILI (2019); MÜLLER/GUIDO (2017); RASHID (2017) – nutzen Python als Programmiersprache für die Erläuterung von KI-Techniken.
- Python bietet die Nutzung von zahlreichen kostenfreien Modulen an, um Funktionserweiterungen vorzunehmen. Insbesondere das Modul OWLready2 stellt einen zentralen Vorteil für den Einsatz der Programmiersprache Python dar, um ein ontologiegestütztes CBR-System als Cloud-native-Anwendung zu implementieren. Das Modul OWLready2 bietet Methoden zur Verarbeitung von Ontologien mittels Python.
- Python gilt als eine der sichersten Programmiersprachen. In der Studie MEND (2018) wurden die in den letzten 10 Jahren verwendeten Programmiersprachen untersucht. Laut dieser Studie belegt die Programmiersprache C mit fast 50 % aller gemeldeten Schwachstellen den „fragwürdigen“ ersten Platz. Java belegt mit 11,4 % aller gemeldetem Schwachstellen den dritten Platz. Python liegt mit nur 6 % auf dem 5. Platz, gefolgt von C ebenfalls mit 6 % und Ruby mit 4 %.

Auf Basis der genannten Vorteile wird die nachfolgende exemplarische Ausgestaltung der Funktionalitäten eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung mithilfe der Programmiersprache Python vorgenommen.

Im Folgenden wird ein Beispiel für die Implementierung einer Serverless-Funktion mittels Python vorgestellt. Zur sprachlichen Vereinfachung wird auch der Ausdruck „Funktion“ synonym zum vollständigen Begriff der Serverless-Funktion verwendet. Sämtliche ab diesem Kapitel programmierten Funktionen wurden sowohl serverless zur Verfügung gestellt als auch mittels

der Programmiersprache Python programmiert. Der Begriff „Funktion“ ist vom Begriff „Python-Methode“ zu unterscheiden. Python-Methoden sind Methoden, die durch Nutzung von Modulen angeboten werden, wie beispielsweise die Python-Methode `startswith()`, die durch das Python-Modul `String` angeboten wird. Das Python-Modul `String` von Python ist ein eingebautes Modul, das nicht zusätzlich importiert werden muss.

Der nachstehende Quellcode für die Funktion `pruefNUTSCode` besitzt als Übergabeparameter die Variable `NUTSCode`. Beginnt der Inhalt der Variabe `NUTSCode` mit `DE`, dann wird die Meldung zurückgegeben „Das Gebiet liegt in Deutschland“. Die Prüfung erfolgt mit der vordefinierten Python-Methode `startswith()`. Erst durch die Nutzung dieser Python-Methode wird zur Laufzeit der Variable `NUTSCode` der Datentyp „String“ zugewiesen. Sollte die Variable `NUTSCode` nicht mit `DE` beginnen, wird die Meldung zurückgegeben: „Das Gebiet liegt nicht in Deutschland“. Einzeilige Kommentare in Python beginnen mit dem Zeichen # (mehrzeilige Kommentierungen beginnen und enden mit der Zeichensequenz “““).

```
1.  def pruefNUTSCode(NUTSCode): # Funktionsbeginn mit Übergabeparameter "NUTSCode"
2.      if(NUTSCode.startswith("DE")): # IF-Abfrage, ob der NUTSCode mit "DE" beginnt.
3.          return "Das Gebiet liegt in Deutschland" # Rückgabe der Zeichenkette
4.      else:
5.          return "Das Gebiet liegt nicht in Deutschland" # Else-Bereich mit Rückgabe
6.  # Funktionsende
```

Abbildung 108: Quellcode für die Funktion `pruefNUTSCode`

Jede Phase des CBR-Zyklus – Retrieve, Reuse, Revise und Retain – kann als eine eigenständige Funktion implementiert und in der Cloud-Umgebung bereitgestellt werden. Die Gesamtheit der Funktionen bildet den CBR-Zyklus ab, wie in der nachfolgenden Abbildung109 dargestellt ist.

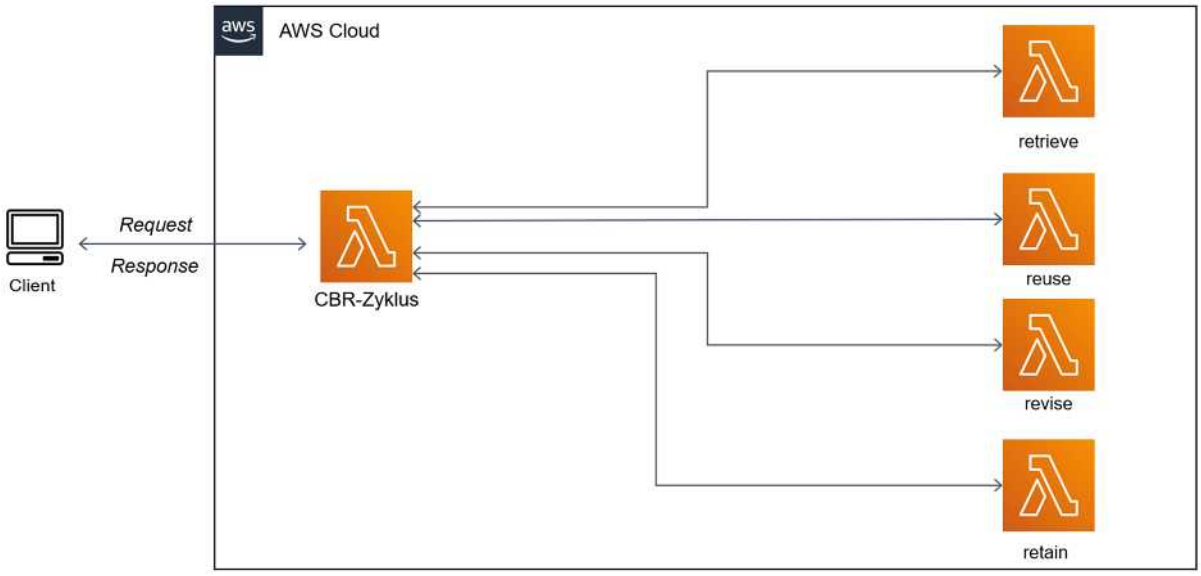


Abbildung 109: Funktionen für den CBR-Zyklus

Bei der Ausführung der Funktion „CBR-Zyklus“ werden die einzelnen Phasen des CBR-Zyklus, die als eigenständige Funktionen implementiert sind, aufgerufen und ausgeführt. Die jeweiligen Ergebnisse werden an die übergeordnete Funktion „CBR-Zyklus“ zurückgegeben, die auf dieser Grundlage die weiteren Berechnungsschritte durchführt.

An späterer Stelle erfolgt beispielhaft eine Implementierung der Retrieve-Phase mit der Ähnlichkeitsberechnung, um die konzeptionelle Gestaltung des Backends zu verdeutlichen. Der Retrieve-Phase wird innerhalb des CBR-Zyklus eine besondere Bedeutung zugemessen, weil der Ähnlichkeitsalgorithmus in der Retrieve-Phase eine hohe Rechenlast verursacht und eine komplexe Berechnung darstellt. Durch die Aufteilung des Ähnlichkeitsalgorithmus in verschiedene Funktionen und die Auslagerung in eine Cloud-Umgebung kann eine Skalierung hinsichtlich der für die einzelnen Funktionen zur Verfügung stehenden Ressourcen erfolgen. Durch die weitere Unterteilung können die Vorteile einer Cloud-Umgebung für einzelne Funktionen genutzt werden, um z. B. die Skalierbarkeit einzelner Berechnungsteile zu ermöglichen. Die folgenden Abbildung110 bis 112 illustrieren das Zusammenspiel der Ähnlichkeitsfunktionen, die in der Retrieve-Phase gemeinsam die Ähnlichkeitsberechnung für ein ontologiegestütztes CBR-System implementieren.

Die Abbildung110 zeigt die Funktion zur Berechnung der vollständigen Ähnlichkeit zwischen zwei Instanzen ($isim$). Sie besteht aus der Funktion zur Bestimmung der Konzeptähnlichkeit ($ksim$) und der Funktion zur Bestimmung der partiellen Instanzähnlichkeit ($esim$). Oberhalb eines Pfeils sind die Eingabeparameter für die betroffene Funktion und unterhalb eines Pfeils die Berechnungsergebnisse der Funktion (Ausgabeparameter) im JSON-Format dargestellt.

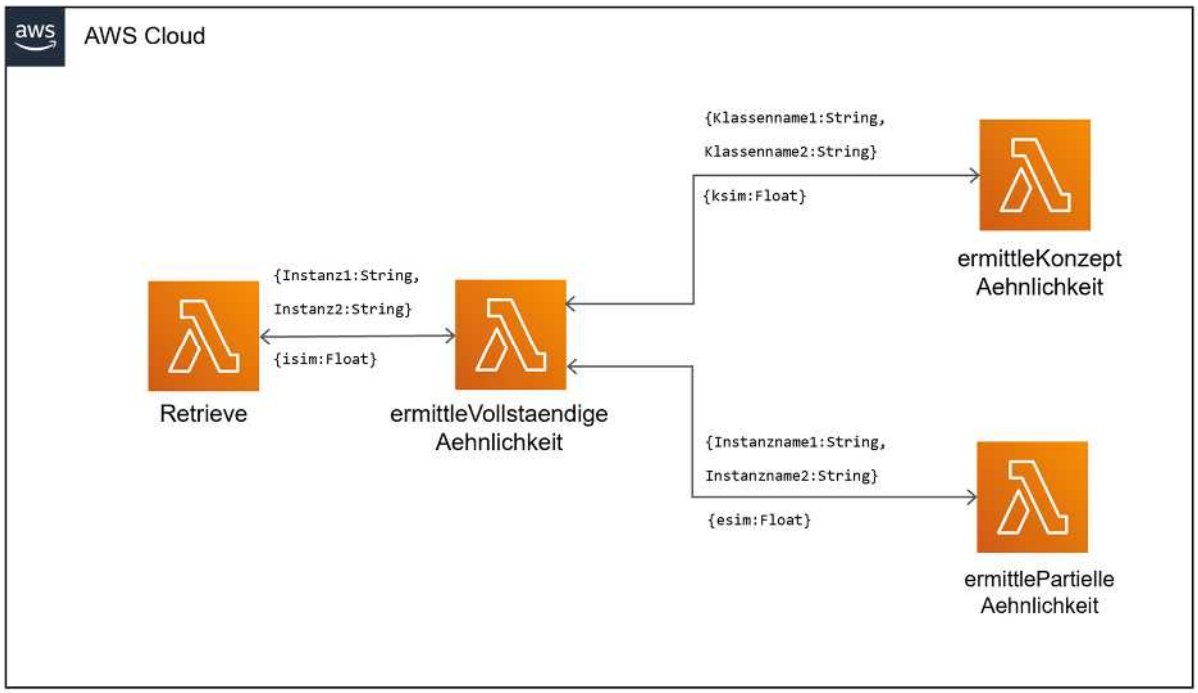


Abbildung 110: Funktionen zur Ähnlichkeitsberechnung

Die nachfolgende Abbildung111 zeigt die weitere Unterteilung in Funktionen zur Bestimmung der Konzeptähnlichkeit, bestehend aus der Funktion zur Berechnung der semantischen Distanz (`ermittleSemantischeDistanz`) und der Funktion zur Bestimmung der Klasseneigenschaften einer Klasse (`ermittleAehnlichkeitKonzepteigenschaft`).

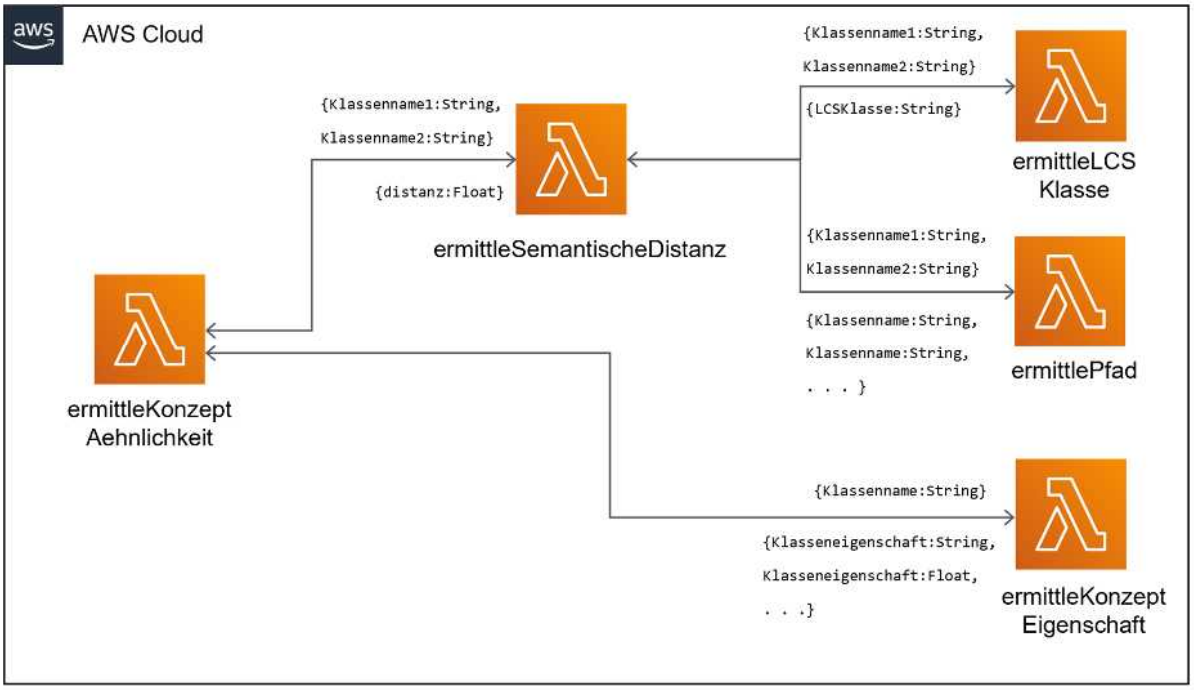


Abbildung 111: Funktionen zur Berechnung der Konzeptähnlichkeit

Die nachfolgende Abbildung 112 illustriert die weitergehende Unterteilung in Funktionen für die Ermittlung der partiellen Instanzenähnlichkeit. Beispielhaft sind drei Funktionen implementiert. Sie berechnen die Ähnlichkeit von Instanzeigenschaften in Bezug auf drei exemplarische Ähnlichkeitstypen: CPV-Code, NUTS-Code und Stringwerte. Diese Funktionen zur Bestimmung der Ähnlichkeit von Instanzeigenschaften werden nicht nur im Amazon Web Service, sondern auch auf der Google Cloud Platform zur Verfügung gestellt.

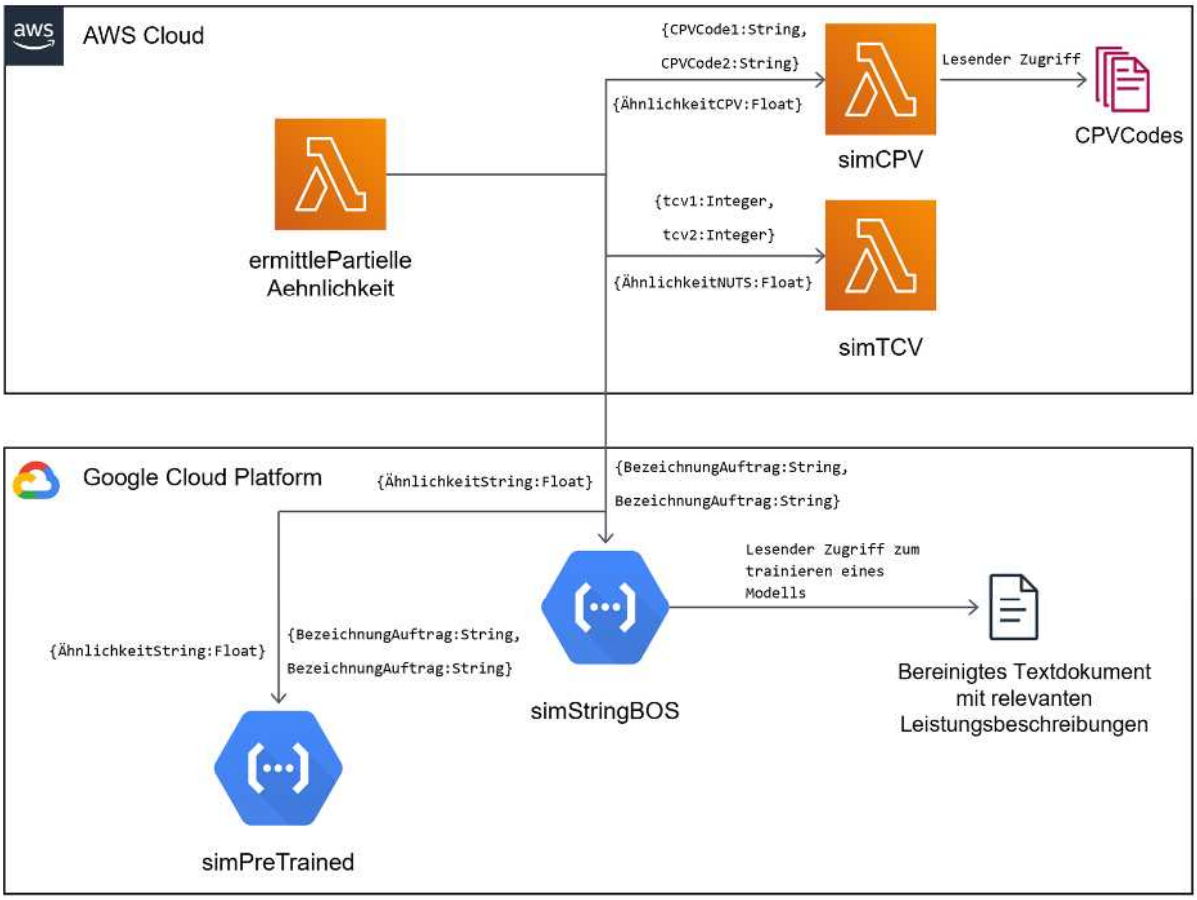


Abbildung 112: Funktionen zur Berechnung der partiellen Instanzenähnlichkeit

Die in den Abbildung110 bis 112 dargestellten Funktionen werden in einem späteren Kapitel ausführlich erläutert. Hierbei wird der Schwerpunkt auf die Funktion `simStringBOS` gelegt werden. Diese Funktion erweist sich als besonders interessant, `weil sie` auf der von Google entwickelten Word2Vec-Technik basiert und Ansatzpunkte für weitergehende Nutzungsmöglichkeiten bietet. Hierauf wird später detailliert eingegangen. Die in Abbildung112 für die Funktion `simStringBOS` angegebenen Eingabeparameter {BezeichnungAuftrag: String, BezeichnungAuftrag: String} sind nur beispielhaft gewählt. Mit dieser Funktion können beliebige Instanzeigenschaften, die auf dem Datentyp „String“ basieren, miteinander verglichen werden.

Die Methoden oder Module einer Programmiersprache reichen für die Bearbeitung einer Ontologie nicht immer aus. In der Programmiersprache Python existiert jedoch seit dem Jahr 2017 ein Modul, das die Konstruktion, Manipulation und Verarbeitung von Ontologien unterstützt. Es handelt sich um das Modul „OWLReady2“, das derzeit in der Version 2-0.39 vorliegt (Stand 30.10.2022) und unter der GNU-LGPL-Lizenz kostenfrei genutzt werden kann. OWLready2 wurde im Forschungslabor „laboratoire d'informatique médicale et d'ingénierie des connaissances en e-Santé“ (LIMICS) der Universität Paris von LAMY entwickelt; vgl. LAMY (2023), S. 1. Das Modul wird in verschiedenen aktuellen Publikationen für die Verarbeitung von Ontologien genutzt. Vgl. beispielsweise DI MARTINO et al. (2022), S. 431; GUSKOV et al. (2022),

S. 368; SARKER et al. (2021), S. 78. In STACK OVERFLOW – einer der bekanntesten Online-Communitys für Entwickler – ist ein eigener Bereich für OWLReady2-bezogene Problemstellungen vorhanden, um Hilfestellung bei Entwicklungsproblemen zu erhalten; vgl. STACK OVERFLOW (2023).

Das Modul *OWLReady2* erlaubt die objektorientierte Implementierung von Ontologien; vgl. LAMY (2021), S. 5. Darüber hinaus ermöglicht das Modul das Laden großer Ontologien. Es kann Ontologien laden, die mehrere hundert Gigabyte groß sind; vgl. LAMY (2017), S. 23. Darüber hinaus wird der Zugriff auf Ontologie-Komponenten mittels einer speziellen Suchmethode gestattet.

Um das Modul OWLReady2 für die Programmierung mit Python einsetzen zu können, muss es zuvor installiert werden. Dazu wird das Modul-Verwaltungsprogramm „pip3“ eingesetzt. Es ermöglicht das automatisierte Herunterladen, Installieren und Aktualisieren von Python-Modulen aus dem Python-Repository, genannt Python Package Index (PyPI), über das Internet. Die Shell-Kommandozeile für die Installation eines Python-Moduls sieht wie folgt aus:

```
1.  pip3 install -U name_des_moduls_dass_installiert_werden_soll
```

Dieses Verwaltungsprogramm kann in der Shell-Kommandozeile unter Unix/Mac oder in der Eingabeaufforderung bei Windows-Systemen ausgeführt werden. Die Befehlszeile installiert ein beliebiges Python-Modul. Sollte das Modul bereits vorhanden sein, wird das Modul aktualisiert. Die Installation des Moduls OWLReady2 erfolgt mit folgender Kommandozeile.

```
1.  pip3 install -U owlready2
```

Die Kommandozeile kann direkt in der integrierten browserbasierten Entwicklungsumgebung ICloud9 von Amazon Web Service ausgeführt werden. Die nachfolgende Abbildung113 veranschaulicht den Download- und Installationsprozess, der bei der Ausführung der oben angeführten Kommandozeile in der Entwicklungsumgebung ICloud9 von Amazon Web Service automatisiert abläuft:

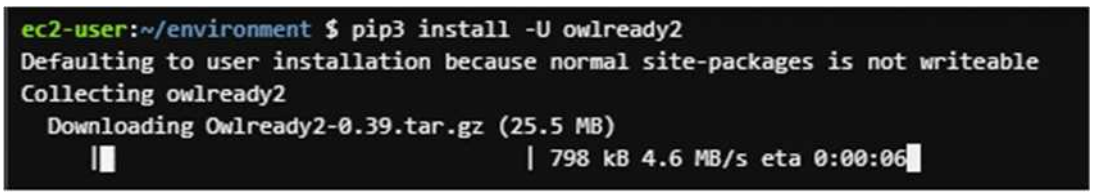


Abbildung 113: Installierung des Moduls OWLReady2

Nach Abschluss der Installation erfolgt eine Meldung, dass das Modul OWLReady2 erfolgreich installiert wurde:

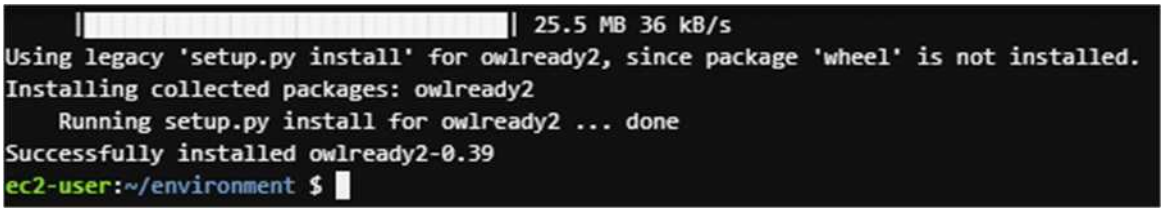


Abbildung 114: Erfolgreiche Installierung des Moduls OWLReady2

Nach der erfolgreichen Installierung kann das Modul OWLready2 im Quellcode genutzt werden. Es gibt zwei Wege, ein Modul im Quellcode zu importieren. Ein erster Weg besteht in der direkten Nutzung der Methoden von OWLReady2. Der Import für die direkte Nutzung der Methoden sieht wie folgt aus:

```
1. from owlready2 import *
2. # Zugang aller Methoden von owlready2 über den direkten Weg
3. onto = get_ontology("//....PfadDerOntologie....").load()
```

Der zweite Weg erfolgt durch die Nutzung der Methoden über die Modul-Bezeichnung OWLReady2:

```
1. import owlready2
2. # Nutzung über owlready2
3. onto = owlready2.get_ontology("//....PfadDerOntologie....").load()
```

Grundsätzlich sind beide Wege möglich. Für die Verständlichkeit des Codes wird jedoch der erstgenannte, direkte Weg bevorzugt. Die Dokumentationen zu OWLReady2 empfehlen ebenfalls diese Art des Imports; vgl. LAMY (2021), S. 83. Dieser Empfehlung wird auch in diesem Beitrag gefolgt.

Das Modul *Gensim* ist ein Open-Source-basiertes Python-Modul, das mehrere Algorithmen zur Analyse großer Textdokumente bereitstellt; vgl. ŘEHŮŘEK (2022). Beispielsweise lässt sich die semantische Struktur eines Dokuments mithilfe von Verfahren des maschinellen Lernens automatisch analysieren. Der Name Gensim leitet sich von „Generate Similar“ ab. Das Modul wurde im Jahr 2008 als Sammlung von Python-Skripten für das Projekt der tschechischen digitalen Mathematikbibliothek (dml.cz) entwickelt. Vgl. ŘEHŮŘEK/SOJKA (2010), S. 48-49; SOJKA (2009), insbesondere S. 75-76. Die Python-Skripte dienten dazu, eine kurze Liste der ähnlichsten mathematischen Artikel zu einem bestimmten Artikel zu erstellen; vgl. ŘEHŮŘEK/SOJKA (2010), S. 47. ŘEHŮŘEK implementierte auf dieser Basis dessen ein eigenständiges Python-Modul und entwickelte die Skripte im Rahmen seiner Dissertation weiter; vgl. ŘEHŮŘEK (2011), S. 19-34, 37-66 u. 67-78. Das Modul Gensim gilt inzwischen als robustes und häufig verwendetes Modul zur automatischen semantischen Analyse von Texten; vgl. EL-AMIR/HAMDY (2020), S. 53; SARKAR (2019), S. 255. Gensim wird in einer Vielzahl von Projekten eingesetzt; vgl. beispielsweise NUGROHO et al. (2023), S. 294; PANDAY/SAHU (2023), S. 294-295; TAYLOR/DU PREEZ (2023), S. 539 u. 550. Gensim steht als öffentliches Projekt dar, das für Weiterentwicklungen offensteht. Die aktuelle Gensim-Version 4.3.0 (Stand 21.12.2022) wurde im Dezember 2022 veröffentlicht; vgl. GITHUB (2022a).

Das Modul Gensim stellt verschiedene Submodule bereit und besitzt bereits vortrainierte Modelle, die für die Berechnung der Ähnlichkeit zwischen Begriffen genutzt werden können. Eine Übersicht der vortrainierten Modelle befindet sich auf GITHUB (2022b). In diesem Beitrag wird ausschließlich das Submodul Word2Vec von Gensim genutzt.

Der Begriff „Training“ wird im Folgenden näher erläutert. Gensim liefert vortrainierte Modelle, die z. B. auf Basis der gesamten Wikipedia (Stand 2017 in englischer Sprache) vortrainiert wurden und ohne erneutes Training für Textanalysen verwendet werden können. Dies erspart den gesamten Prozess des Trainings von Textdokumenten. Weitere erwähnenswerte Modelle sind z. B. das vortrainierte Modell von Google, das auf Basis von Google News (mit einem Umfang von 100 Milliarden Wörtern) vortrainiert wurde. Darüber hinaus existieren weitere vortrainierte Modelle für verschiedene Sprachen, z. B. Deutsch, Chinesisch und Französisch. In diesem Beitrag wird neben einem selbsttrainierten Modell auf Basis von (zuvor anonymisierten) Leistungsbeschreibungen aus sicherheitskritischen IT-Projekten auch ein vortrainiertes Modell verwendet, um die unterschiedlichen Ergebnisse vergleichend bewerten zu können. Insbesondere die größeren Modelle, wie z. B. von Google, benötigen (trotz ihres Vortrainings) eine hohe Rechenkapazität, um eine Ähnlichkeitsberechnung während der Laufzeit der Programmausführung durchführen zu können. Diese Kapazität kann zwar zur Laufzeit durch eine Cloud-Umgebung bereitgestellt werden, verursacht aber höhere Kosten, da mehr Leistung verbraucht wird. Dies stellt eine nicht zu vernachlässigende Einschränkung bei der Verwendung großer vortrainierter Modelle dar. Darauf wird später zurückgekommen.

Darüber hinaus ist darauf hinzuweisen, dass im Zusammenhang mit der Ähnlichkeitsberechnung mittels des Moduls Gensim streng genommen keine Begriffe (im semantischen Sinn), sondern Worte (im syntaktischen Sinn) gemeint sind. Auf diese Unterscheidung wird später präzisierend zurückgekommen. Allerdings werden im Folgenden die Bezeichnungen „Begriff“ und „Wort“ der Einfachheit halber zunächst synonym verwendet.

Der Import des Moduls Gensim und seines Submoduls Word2Vec kann auf der Google Cloud Platform ohne vorherige Installierung durchgeführt werden. Der Import des Moduls erfolgt mithilfe folgenden Befehls:

```
1.  # Importiere von dem Gensim Modul ausschließlich das Word2Vec Submodul
2.  from gensim.models import Word2Vec
```

Word2Vec ist eine Wordeinbettungs-Technik (word embedding technique), die von MIKOLOV im Jahr 2013 im Rahmen seiner Tätigkeit bei Google publiziert wurde und seitdem öffentlich für alle Nutzer zur Verfügung steht. Die grundlegenden Forschungen zu einer vektoriellen Darstellung von Wörtern wurden von MIKOLOV bereits während seiner Tätigkeit bei Microsoft durchgeführt. Sie wurden in der Publikation MIKOLOV/YIH/ZWEIG (2013) veröffentlicht, die als zentrale Grundlage für Word2Vec gilt. Weitere relevante Arbeiten zu diesem Thema sind MIKOLOV et al. (2013a) und MIKOLOV et al. (2013b).

Die Word2Vec-Technik wandelt Text in Wortvektoren um, um die „Semantik“ von Wörtern und die Beziehung zwischen Wörtern zu erfassen sowie auf dieser Grundlage Ähnlichkeiten

zwischen Wörtern zu berechnen. Das Ziel von Word2Vec ist es, Vektoren ähnlicher Wörter in einem Vektorraum zu gruppieren, um den Kontext zu erkennen; vgl. MIKOLOV et al. (2013a), S. 2; MIKOLOV/YIH/ZWEIG (2013), S. 746.

Die Diskussion darüber, ob die Word2Vec-Technik tatsächlich die „Semantik" von Wörtern erfassen kann, ist umstritten. Streng genommen erfasst Word2Vec lediglich syntaktische Koinzidenzen von Wörtern in Textkorpora, lernt daraus statistische Beziehungen zwischen Wörtern und nutzt sie, um in unvollständigen Texten die – aus statistischer Sicht – wahrscheinlichsten Textergänzungen mit neuen Wörtern abzuleiten. Dies kann nicht als echtes „semantisches" Verständnis betrachtet werden, sondern beruht auf einer „ausgefeilten" Analyse der quantitativ-statistischen – und somit rein syntaktischen – Korrelationen zwischen Wörtern. Dies wird beispielsweise anhand der Herausforderungen deutlich, vor denen Word2Vec bei der Unterscheidung zwischen mehreren Bedeutungen eines Wortes (Polysemie) und bei der Identifizierung unterschiedlicher Wörter, die gleich geschrieben werden (Homonymie), steht. Beide Aspekte können zu fehlerhaften „semantischen" Begriffsrepräsentationen in Word2Vec führen. Trotz dieser „Semantikzweifel" und ihrer gut begründeten Diskutierbarkeit wird nachfolgend hinsichtlich der Word2Vec-Technik weiterhin von „Semantik" gesprochen. Dies wird damit begründet, dass die Word2Vec-Technik Analogieschlüsse ziehen kann und die Erstellung von Wortvektoren ermöglicht, bei denen (syntaktisch) ähnliche Wörter mittels ähnlicher Vektoren repräsentiert werden. Obwohl dies keine umfassende semantische Erkenntnis von Begriffsinhalten im semiotischen Sinn darstellt, spiegelt es dennoch die Fähigkeit wider, Bedeutungsnuancen und Beziehungen zwischen Wörtern in einem abstrakten Vektorraum zu modellieren. Darauf wird an späterer Stelle vertiefend zurückgekommen.

Die Word2Vec-Technik nutzt ein zweischichtiges Künstliches Neuronales Netz, um ein Modell (darauf wird später näher eingegangen) zu berechnen; vgl. MIKOLOV/YIH/ZWEIG (2013), S. 746; MIKOLOV et al. (2013b), S. 1. Durch das überwachte Lernen (supervised learning) steigert Word2Vec seine Fähigkeit, Zusammenhänge zu erkennen und auszugegeben. Die Darstellung von Wörtern durch Vektoren ermöglicht es, Zusammenhänge durch einfache mathematische Operationen zu erkennen. Ein oftmals zitiertes Beispiel von MIKOLOV et al. (2013a), S. 2, und MIKOLOV/YIH/ZWEIG (2013), S. 748-749 (ähnlich auch in MIKOLOV et al. (2013b), S. 1), lautet:

$$\text{vec}(\text{„King"}) - \text{vec}(\text{„Man"}) + \text{vec}(\text{„Woman"}) \sim \text{vec}(\text{„Queen"})$$

Die Erkenntnis aus solchen Untersuchungen ist, dass sich durch die Modellierung von Wortzusammenhängen mittels Vektoren syntaktische und „semantische" (s. o.) Beziehungen ergeben, die mittels mathematischer Operationen analysiert werden können; vgl. MIKOLOV et al. (2013a), S. 2. Die Grundlagen der Word2Vec-Technik sind von entscheidender Bedeutung für die jüngsten Fortschritte in der computerbasierten Verarbeitung von Texten; vgl. DEVLIN et al. (2019), S. 4171-4173; HOWARD/RUDER (2018), S. 328; RONG (2014), S. 1.

Die Word2Vec-Technik bietet zwei Berechnungsmethoden für das Lernen von Worteinbettungen mithilfe eines Künstlichen Neuronalen Netzes. Die Methoden berechnen jeweils ein Modell, mit dem ein Zentrumswort basierend auf den angrenzenden Kontextwörtern vorausgesagt

werden kann oder mehrere Kontextwörter basierend auf einem Eingabewort herausgegeben werden können. Die nachfolgende Abbildung 115 zeigt die beiden Modelle, die mithilfe der Word2Vec-Technik berechnet werden können.

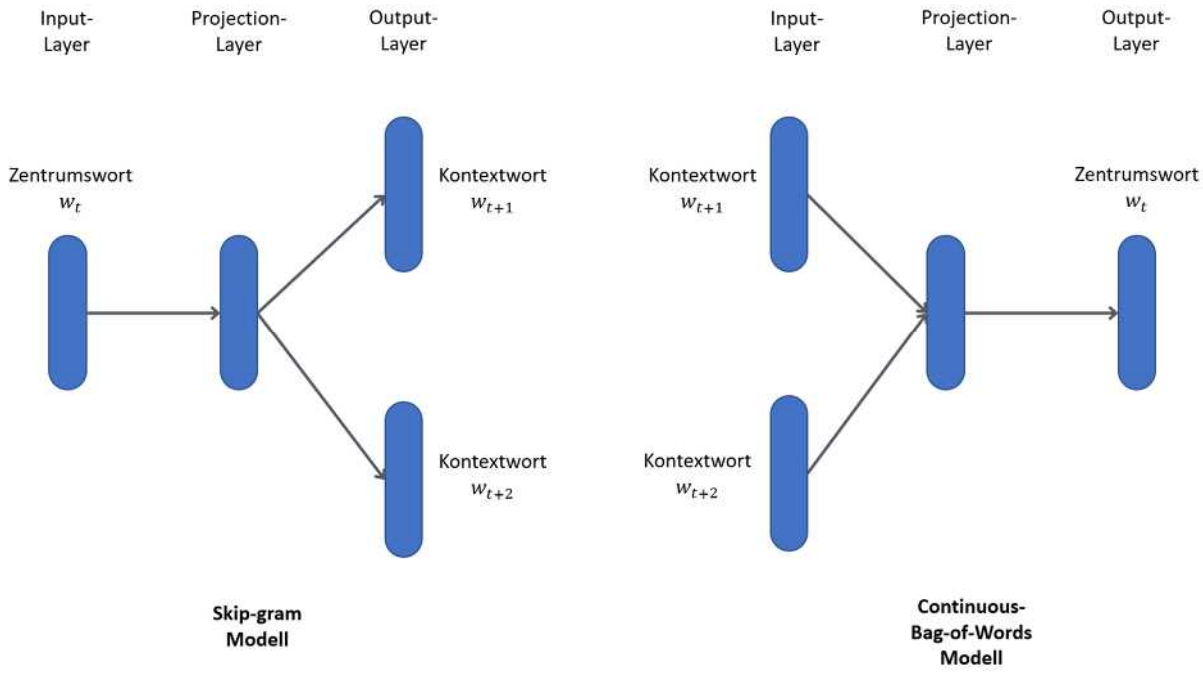


Abbildung 115: Word2Vec-Modelle

Die beiden Berechnungsmethoden, die von der Word2Vec-Technik bereitstellt werden, heißen:

- Continuous-Bag-of-Words-Modell (CBOW-Modell) und
- Skip-gram-Modell (SG-Modell).

Beim CBOW-Modell wird das gesuchte Wort (Zentrumswort) auf Basis von angrenzenden Kontextwörtern vorhergesagt; vgl. MIKOLOV et al. (2013a), S. 5. Die angrenzenden Kontextwörter bestehen aus Wörtern, die sich vor oder nach dem gesuchten Zentrumswort befinden. Die Reihenfolge der Kontextwörter ist bei diesem Modell nicht relevant. Die Stärke des CBOW-Modells liegt darin, dass das Modell syntaktische Beziehungen zwischen zwei Wörtern besser erfassen kann als das Skip-gram-Modell; vgl. MIKOLOV et al. (2013a), S. 7.

Beim Skip-gram-Modell werden Kontextwörter auf Basis eines eingegebenen Zentrumsworts vorhergesagt; vgl. MIKOLOV et al. (2013a), S. 5. Im Grunde genommen arbeitet dieses Modell entgegengesetzt zum CBOW-Modell. Der Vorteil des Skip-gram-Modells erstreckt sich darauf, dass das Skip-gram-Modell „semantische“ Beziehungen zwischen zwei Wörtern besser erkennen kann als das CBOW-Modell; vgl. MIKOLOV et al. (2013a), S. 7. Des Weiteren verbraucht das Skip-gram-Modell laut MIKOLOV et al. (2013a), S. 9, beim Trainieren eines Modells weniger CPU-Ressourcen als das CBOW-Modell. Daher wird das Skip-gram-Modell den nachfolgenden Erläuterung einer Ähnlichkeitsberechnung zugrunde gelegt.

Das Ziel der folgenden Erläuterungen ist es, die Ähnlichkeitsberechnung mithilfe der Word2-Vec-Technik auf der Grundlage eines Skip-gram-Modells anhand eines Beispielsatzes zu erläutern. Einige Parameter (im Folgenden als Hyperparameter bezeichnet), die nachfolgend erläutert werden, sind bei der späteren Implementierung der Funktion `simStringBOS` von Bedeutung. Bei der Verwendung einzelner Methoden der Word2Vec-Technik müssen zunächst verschiedene Parameter gesetzt werden. Dies setzt ein Grundverständnis der Berechnungen im zugrundeliegenden Künstlichen Neuronalen Netzes voraus.

Die nachfolgenden Erläuterungen und Berechnungen beziehen sich auf folgenden Beispielsatz aus SENATSVERWALTUNG FÜR INNERES UND SPORT (2016), S. 2:

„Die Polizei Berlin und die Feuerwehr Berlin
betreiben jeweils unabhängig voneinander Leitstellen.“

Der Beispielsatz stellt gleichzeitig auch den hier betrachteten Textkorpus dar. Dies wird hier vereinfachend unterstellt, obwohl ein Textkorpus in der Regel aus einem umfangreichen Textdokument und nicht nur aus einem einzelnen Satz besteht. Dieser Textkorpus (Beispielsatz) Z umfasst zwar zwölf Wörter (dargestellt als w_i, wobei i die Position im Textkorpus bezeichnet), aber nur zehn individuelle Wörter, weil die Wörter „Berlin“ und „die“ im Textkorpus jeweils doppelt vorkommen. Die individuellen Wörter, die unabhängig von ihrer Schreibweise nur einmal im Textkorpus vorkommen, lauten:

1. „Die“ und „die“
2. „Polizei“
3. „Berlin“
4. „und“
5. „Feuerwehr“
6. „betrieben“
7. „jeweils“
8. „unabhängig“
9. „voneinander“
10. „Leitstellen“

Die individuellen Wörter definieren das Vokabular V des Textkorpus Z. Die Anzahl der Elemente des Vokabulars wird mit $|V| = 10$ dargestellt. Dabei gilt es zu beachten, dass im Textkorpus jedes Wort einmal gezählt wird, und zwar unabhängig von seiner Form (z. B. Singular versus Plural). Dies bedeutet, dass „Leitstelle“ und „Leitstellen“ als zwei verschiedene Wörter im Vokabular betrachtet werden. Jedoch hat das Skip-gram-Modell in der Regel den Vorteil, mit solchen Problemfällen umgehen zu können, bei denen verschiedene Formen eines Wortes vorkommen. Denn es versucht, semantisch ähnliche Wörter zu identifizieren. Daher wird es „wahrscheinlich“ lernen, dass „Leitstelle“ und „Leitstellen“ in ähnlichen Kontexten auftreten

und daher ähnliche Wortvektoren besitzen. Jedoch erkennt Word2Vec nicht von „Natur“ aus, dass es sich dabei um denselben Begriff handelt, es sei denn, dies wird während des Trainings speziell berücksichtigt, möglicherweise durch manuelle oder teilautomatische Aufbereitung der Trainingsdaten.

Die Analyse des Textkorpus erfolgt mittels eines Fensters („window size“), das mit einer festen Größe m (beispielsweise $m = 1$) angegeben wird. Die Fenstergröße mit $m = 1$ stellt zwar kein realistisches Anwendungsszenario dar. Diese kleine Fenstergröße wurde aber als Beispiel gewählt, um die Komplexität der manuellen Berechnung zu reduzieren und eine Erläuterung anhand eines einfachen Beispiels vornehmen zu können. In der Regel werden größere Fenstergrößen bevorzugt, um einen breiteren Kontext für die Wortvektorisierung zu erfassen. Ein größeres Fenster m kann zu mehr Trainingsbeispielen und zu einer höheren Genauigkeit der Ähnlichkeitsberechnung führen, allerdings auf Kosten der Trainingszeit; vgl. MIKOLOV et al. (2013b), S. 8. Die Wahl der „optimalen“ Fenstergröße stellt ein gesondertes Problem dar, auf das im weiteren Verlauf dieses Beitrags eingegangen wird.

Das Wort w_t in der Mitte des Fensters wird als Zentrumswort („center-word“ oder „target-word“) bezeichnet. Die vorangegangenen (w_{t-m}) und die nachfolgenden (w_{t+m}) Wörter heißen Kontextwörter. Das Fenster mit der Größe m durchläuft den gesamten Satz, um ein Wortpaar als Trainingsmuster für das Modell zu generieren. Dieses Wortpaar besteht aus einem Zentrumswort, das später als Input verwendet wird, und einem seiner Kontextwörter, welches das Ziel im Kontext markiert. Der Kontext meint in diesem Zusammenhang das Kontextwort, auf welches das Modell abzielt, indem es die Beziehung und Wahrscheinlichkeit zwischen dem Zentrumswort und dem Kontextwort in einem Modell ermittelt.

Die nachfolgende Abbildungen 116 bis 118 illustrieren die ersten drei Iterationen zur Extraktion des Trainingsmodells. Das Zentrumswort w_t ist grün dargestellt, die Kontextwörter w_{t-1} und w_{t+1} für das Fenster mit der Größe $m = 1$ werden jeweils grau dargestellt.

1. Iteration

die	polizei	berlin	und	die	feuer-wehr	berlin	betrei-ben	jeweils	unab-hängig	voneinander	leitstellen

Abbildung 116: Erste Iteration

Das Trainingsmuster lautet: *(die, polizei)*

2. Iteration

die	polizei	berlin	und	die	feuerwehr	berlin	betreiben	jeweils	unabhängig	voneinander	leitstellen

Abbildung 117: Zweite Iteration

Das Trainingsmuster lautet: *(polizei, die), (polizei, berlin)*

3. Iteration

die		polizei	berlin	und	die	feuerwehr	berlin	betrei-ben	jeweils	unabhängig	voneinan-der	leitstellen

Abbildung 118: Dritte Iteration

Das Trainingsmuster lautet: *(berlin, polizei), (berlin, und)*

Für die nachfolgenden Erklärungen wird exemplarisch das Zentrumswort „feuerwehr“ mit den Kontextwörtern „die“ und „berlin“ gewählt. Dies stellt die 6. Iteration dar.

6. Iteration

die		polizei	berlin	und	die	feuerwehr	berlin	betrei-ben	jeweils	unabhängig	voneinan-der	leitstellen

Abbildung 119: Sechste Iteration

Das Trainingsmuster lautet: *(feuerwehr, die), (feuerwehr, berlin).*

Zunächst werden das Zentrumswort und die Kontextwörter als One-Hot-Kodierung in einem Vektor dargestellt. Bei der One-Hot-Kodierung wird ein Merkmal mit einer binären Variable (1 oder 0) dargestellt; vgl. KULKARNI/SHIVANANDA (2021), S. 64; BISONG (2019), S. 336. Durch die One-Hot-Kodierung wird eine maschinelle Verarbeitung möglich. In dem Textkorpus wird das Vorkommen des Wortes mit einer 1 dargestellt und das Nichtvorkommen mit einer 0. One-Hot-Kodierungen werden häufig in den Bereichen des maschinellen Lernens eingesetzt; vgl. KULKARNI/SHIVANANDA (2021), S. 64. Hier wird eine weitere Einschränkung deutlich: Bei einem großen Textkorpus wird ein großer Vektor benötigt, um jedes Wort des Textkorpus als One-Hot-Kodierung auszudrücken. Dies führt zu einer höheren Rechenlast, die zwar durch eine Cloud-Umgebung abgedeckt werden kann, aber bei großen Textkorpora nicht vernachlässigt werden darf.

Die nachfolgende Tabelle 61 illustriert die One-Hot-Kodierung für alle Wörter im Vokabular.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Wort	die	polizei	berlin	und	feuerwehr	betreiben	jeweils	unabhängig	voneinander	leitstellen
die	1	0	0	0	0	0	0	0	0	0
polizei	0	1	0	0	0	0	0	0	0	0
berlin	0	0	1	0	0	0	0	0	0	0
und	0	0	0	1	0	0	0	0	0	0
feuerwehr	0	0	0	0	1	0	0	0	0	0
betrieben	0	0	0	0	0	1	0	0	0	0
Jeweils	0	0	0	0	0	0	1	0	0	0
unabhängig	0	0	0	0	0	0	0	1	0	0
voneinander	0	0	0	0	0	0	0	0	1	0
leitstellen	0	0	0	0	0	0	0	0	0	1

Tabelle 61: One-Hot-Kodierung des Textkorpus *Z*

In der Tabelle61 repräsentiert jede Zeile ein Wort aus dem Vokabular. Die Spalten stellen die verschiedenen Wörter im Vokabular dar und geben die Position jedes Wortes in der Vokabularliste an. Eine 1 in einer Zelle zeigt an, dass das Wort in dieser Zeile an der betroffenen Position im Vokabular vorhanden ist. Eine 0 zeigt hingegen an, dass das Wort in dieser Zeile an der betroffenen Position im Vokabular nicht vorhanden ist.

Anhand der Tabelle61 lässt sich der Vektor eines Wortes aus dem Vokabular als One-Hot-Kodierung ablesen (hellblau dargestellt). So lautet der Vektor als One-Hot-Kodierung zum Beispiel für das Wort „Polizei“ wie folgt:

$$\overrightarrow{polizei} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Die Dimension des Vektors beträgt $10\ x\ 1$, da zehn individuelle Wörter im Vokabular enthalten sind.

Für die nachfolgenden Berechnungen sind die Vektoren des Zentrumsworts und der Kontextwörter – jeweils in One-Hot-Kodierung – von Bedeutung, da sie in die Berechnung als Schlüsselkomponenten einfließen. Die Vektoren des Zentrumswortes „feuerwehr“ (dargestellt als $\vec{x}$), des Kontextwortes „die“ (dargestellt als $\overrightarrow{y_1}$) und des Kontextwortes „berlin“ (dargestellt als $\overrightarrow{y_2}$) werden wie folgt in einer One-Hot-Kodierung dargestellt:

$$\vec{x} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \qquad \overrightarrow{y_1} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \qquad \overrightarrow{y_2} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

Die Vektoren $\overrightarrow{y_1}$ und $\overrightarrow{y_2}$, welche die Kontextwörter repräsentieren, werden in der nachfolgenden Berechnung auch als Zielvektoren bezeichnet.

Der Vektor $\vec{x}$ dient als Input für den Input Layer eines Künstlichen Neuronalen Netzes. Bei dem Künstlichen Neuronalen Netz handelt es sich um ein Feed-Forward-Netz. Feed-Forward bedeutet in diesem Zusammenhang, dass innerhalb eines Künstlichen Neuronalen Netzwerks Informationen ausschließlich vorwärts, also an die nächste Schicht von Neuronen weitergeleitet werden; vgl. MATZKA (2021), S. 117. Hinsichtlich der Word2Vec-Technik bedeutet dies, dass Informationen ausschließlich vom Input Layer zu einem oder mehreren Hidden Layer und dann

zum Output Layer weitergeleitet werden. Die Vorgehensweise wird anschließend näher erläutert. Im Gegensatz dazu stehen rekurrente Künstliche Neuronale Netze, die zwar eine Erweiterung des Feed-Forward darstellen, aber in denen Informationen auch an Neuronen der gleichen oder vorherigen Schicht übergeben werden können; vgl. MATZKA (2021), S. 128.

Im hier betrachteten Feed-Forward-Netz existieren neben dem Input Layer ($\vec{x}$) ein Hidden Layer ($\vec{h}$) und ein Output Layer ($\vec{u}$). Das Modell der Word2Vec-Technik besteht im Wesentlichen aus den beiden Gewichtsmatrizen W_{Input} und W_{Output} für die Gewichte der Verbindungen zwischen den Neuronen des Künstlichen Neuronalen Netzes. Die zentrale Aufgabe der Word2-Vec-Technik besteht darin, die Gewichtsmatrizen W_{Input} und W_{Output} während des Trainingsprozesses zu „optimieren“. Dabei versucht das Modell, die Wahrscheinlichkeiten für das Auftreten von Kontextwörtern in Bezug auf ein Zentrumswort zu maximieren, um eine bestmögliche Vorhersage von Kontextwörtern zu ermöglichen. Daher lernt das Modell, wie die Wörter im Vokabular in „semantischer“ – streng genommen in statistischer (und somit syntaktischer) – Hinsicht miteinander in Beziehung stehen, indem es die Gewichtungen in den Matrizen W_{Input} und W_{Output} anpasst. Dies wird im Folgenden detailliert erläutert.

Die nachfolgende Abbildung120 illustriert die Berechnung der Gewichtsmatrizen W_{Input} und W_{Output} für die Gewichte der Verbindungen zwischen den Neuronen des Künstliche Neuronale Netzes. Die folgenden Erläuterungen basieren auf dieser Abbildung.

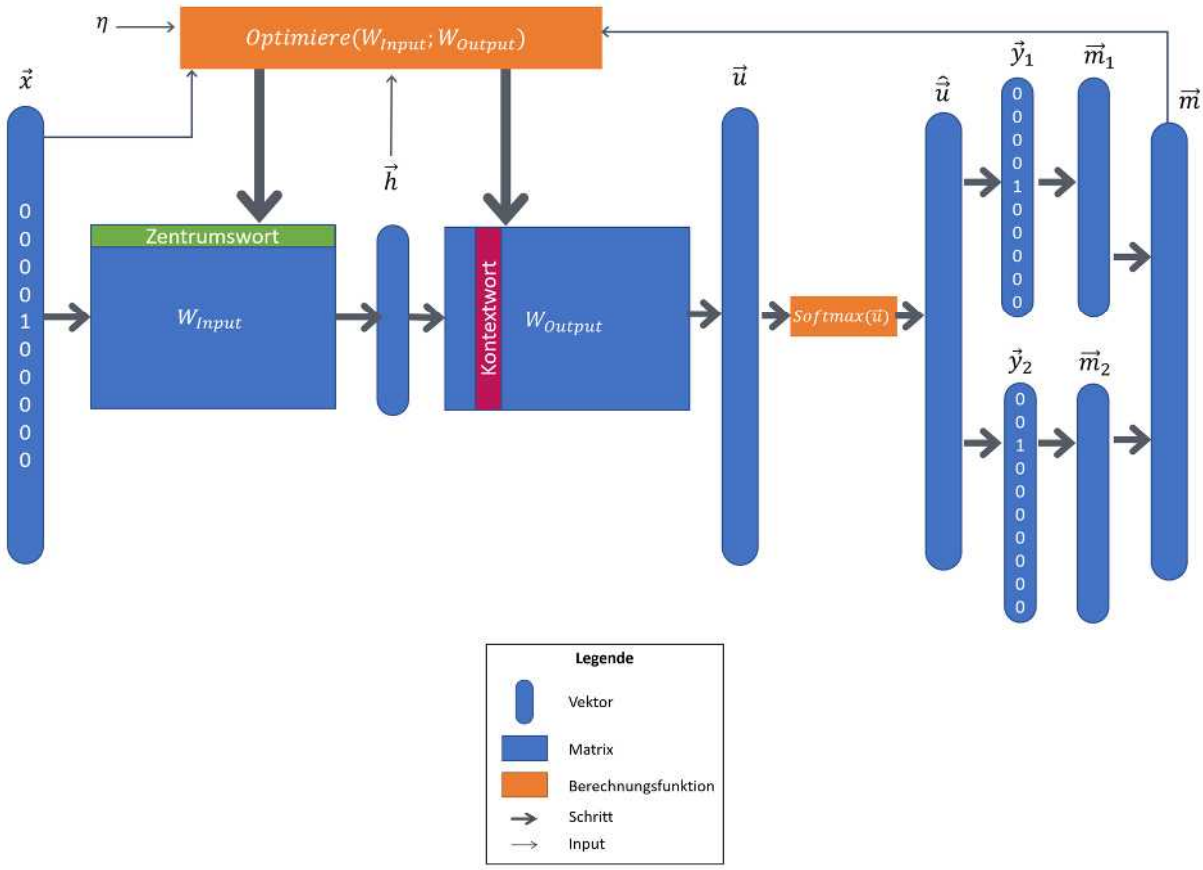


Abbildung 120: Berechnungsschritte für das Skip-gram-Modell

Im nächsten Schritt wird die Variable *N* definiert. Die Variable *N* bezieht sich auf die Dimension des Wortvektors. Nachfolgend wird auch die Bezeichnung *N*-dimensionaler Wortvektor verwendet. Die Dimension N des Wortvektors stellt einen Hyperparameter für den Hidden Layer des Künstlichen Neuronalen Netzes dar. In der Word2Vec-Technik wird der Hyperparameter N als „Size" bezeichnet und wird vor dem Training des Modells festgelegt. Ein Hyperparameter ist ein Parameter in Algorithmen des maschinellen Lernens, der zur Steuerung des Trainingsalgorithmus verwendet wird und sich von anderen Parametern dadurch unterscheidet, dass er vor dem Training des Modells festgelegt werden muss; vgl. AGRAWAL (2021), S. 4-5. Dies gilt auch für die Hyperparameter „Fenster" m und „Lernrate" η. Auf die Lernrate η wird im späteren Verlauf näher eingegangen.

Die Gewichtsmatrizen W_{Input} und W_{Output} enthalten die Wortvektoren als Zeilen oder Spalten. Jede Zeile in W_{Input} und jede Spalte in W_{Output} entspricht einem Wortvektor. Die Anzahl der Zeilen in W_{Input} und die Anzahl der Spalten in W_{Output} entsprechen der Größe mit $|V| = 10$ des Vokabulars *V*; vgl. RONG (2014), S. 8. Siehe dazu auch Abbildung120. Dort ist in der Gewichtsmatrix W_{Input} die Zeile mit dem Wortvektor für das Zentrumswort in Grün hervorgehoben, während in der Gewichtsmatrix W_{Output} die Spalte mit dem Wortvektor für das Kontextwort in Rot hervorgehoben ist. An dieser Stelle wird deutlich, dass die Gewichtsmatrizen zwar aus demselben Vokabular V stammen, aber unterschiedliche Inhalte repräsentieren. Die Gewichtsmatrix W_{Input} umfasst alle Zentrumswörter, die Gewichtsmatrix W_{Output} hingegen alle Kontextwörter.

Im nachfolgenden Beispiel wird für den N-dimensionalen Wortvektor einfachheitshalber die Dimension $N = 3$ gewählt. Der N-dimensionale Wortvektor ist zwar hinsichtlich einer praktischen Anwendung, analog zur Fenstergröße mit $m = 1$, viel zu klein dimensioniert. In praktischen Anwendungen der Word2Vec-Technik liegt die Dimension des Wortvektors in der Regel bei $N = 300$; vgl. MIKOLOV et al. (2013b), S. 6. Die Dimension $N = 3$ wird jedoch zugrunde gelegt, um bei den manuellen Berechnungen keine zu großen Gewichtsmatrizen bewältigen zu müssen.

Die Gewichtsmatrix W_{Input} hat die Dimension von $10\ x\ 3$ und ist wie folgt aufgebaut:

$$W_{Input} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \\ w_{61} & w_{62} & w_{63} \\ w_{71} & w_{72} & w_{73} \\ w_{81} & w_{82} & w_{83} \\ w_{91} & w_{92} & w_{93} \\ w_{101} & w_{102} & w_{103} \end{bmatrix}$$

Die Gewichtsmatrix W_{Output} hat die Dimension von $3\ x\ 10$ und ist wie folgt aufgebaut:

$$W_{Output} = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} & v_{17} & v_{18} & v_{19} & v_{110} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} & v_{26} & v_{27} & v_{28} & v_{29} & v_{210} \\ v_{31} & v_{32} & v_{33} & v_{34} & v_{35} & v_{36} & v_{37} & v_{38} & v_{39} & v_{310} \end{bmatrix}$$

Die Gewichtsmatrizen W_{Input} und W_{Output} werden zu Beginn des Trainingsprozesses mit zufälligen Werten initialisiert. Diese zufällige Initialisierung dient als Ausgangspunkt für das Training und ermöglicht es dem Modell, die Wortvektoren während des Trainings schrittweise zu „optimieren“; vgl. AYYADEVARA (2018), S. 170. Die Werte in den Gewichtsmatrizen W_{Input} und W_{Output} sind maschinenlesbar, können jedoch von Menschen nicht direkt interpretiert werden. Maschinenlesbar bedeutet, dass die Werte von Computern verarbeitet werden können.

Im Folgenden werden die beiden Gewichtsmatrizen W_{Input} und W_{Output} initial mit beliebigen Werten belegt:

$$W_{Input} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix}$$

$$W_{Output} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix}$$

Im nächsten Schritt wird der Vektor $\vec{h}$ des Hidden-Layers berechnet. Der Hidden Layer wird auch als Projektionsschicht bezeichnet, da der Output dieses Layers ein N-dimensionaler Wortvektor ist, der durch den One-Hot-Kodierung-Inputvektor $\vec{x}$ projiziert wird.

In der nachfolgenden Abbildung121, die einen „angepassten“ Ausschnitt aus der Abbildung120 zeigt, wird die Berechnung des Vektors $\vec{h}^T$ veranschaulicht. Dabei ist zu beachten, dass der Vektor $\vec{x}$ die Dimension $10\ x\ 1$ besitzt und für die folgende Berechnung der transponierte Vektor $\vec{x}^T$ verwendet werden muss, damit eine Matrizenmultiplikation mit der Matrix W_{Input}, die eine Dimension von $10\ x\ 3$ hat, durchgeführt werden kann. Durch den transponierten Vektor $\vec{x}^T$, der die Dimension von $1\ x\ 10$ aufweist, kann eine Matrizenmultiplikation mit der Matrix W_{Input} erfolgen, die die Dimension von $10\ x\ 3$ besitzt.

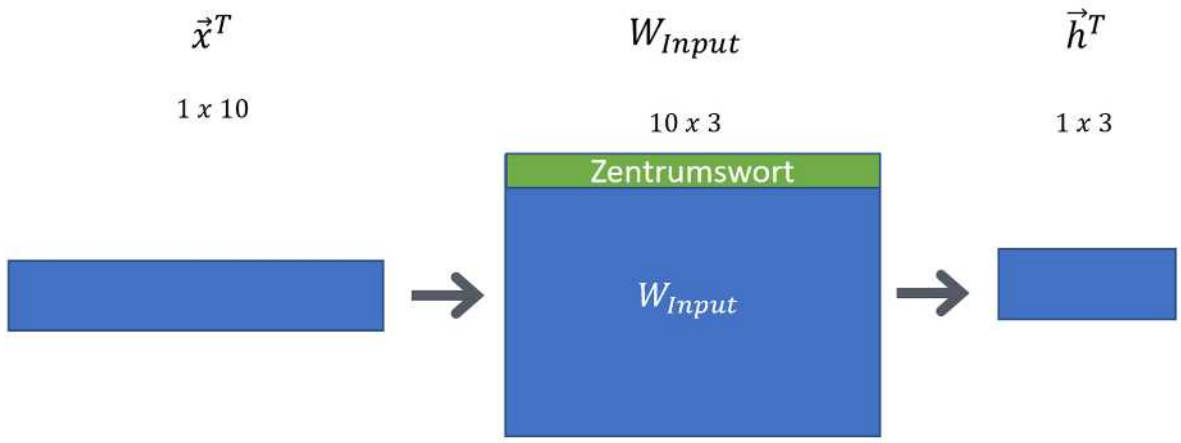


Abbildung 121: Berechnung des Vektors $\vec{h}^T$

Die Berechnung des Vektors $\vec{h}^T$ erfolgt auf folgende Weise:

$$\vec{h}^T = \vec{x}^T * W_{Input} = [h_1 \quad h_2 \quad h_3]$$

$$\vec{h}^T = [0.13 \quad 0.14 \quad 0.15]$$

Bevor der Vorhersagevektor $\hat{\vec{u}}$ berechnet werden kann, ist der Vektor $\vec{u}$ zu berechnen. In der nachfolgenden Abbildung122, die abermals einen „angepassten“ Ausschnitt aus der Abbildung120 zeigt, wird die Berechnung des Vektors $\vec{u}^T$ veranschaulicht:

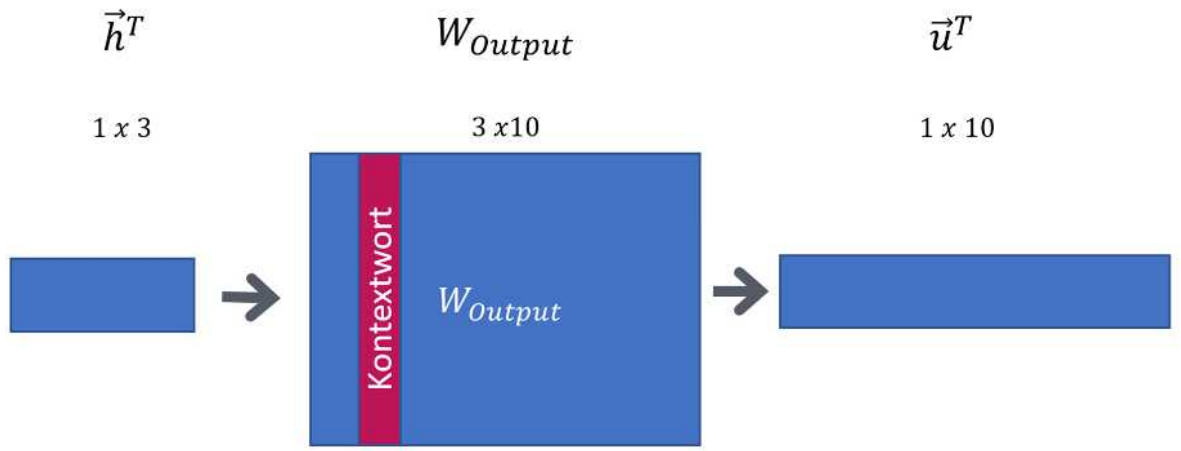


Abbildung 122: Berechnung des Vektors $\vec{u}^T$

Die Berechnung des Vektors $\vec{u}^T$ erfolgt auf folgende Weise:

$$\vec{u}^T = \vec{h}^T * W_{Output} = [u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6 \quad u_7 \quad u_8 \quad u_9 \quad u_{10}]$$

$$\vec{u}^T = [0.06 \quad 0.076 \quad 0.092 \quad 0.108 \quad 0.124 \quad 0.139 \quad 0.155 \quad 0.171 \quad 0.187 \quad 0.203]$$

Mithilfe des Vektors $\vec{u}^T$ erfolgt die Berechnung des Vorhersagevektors $\widehat{\vec{u}^T}$. Um für den Vektor $\vec{u}^T$ eine Wahrscheinlichkeitsverteilung über alle Worte zu erzeugen, wird eine Softmax-Funktion σ verwendet, die den Vektor $\vec{u}^T$ auf Werte zwischen 0 und 1 normalisiert; vgl. GOLDBERG/ LEVY (2014), S. 2; RONG (2014), S. 8. Hierbei stehen 1 für das „sichere Ergebnis", also eine Wahrscheinlichkeit von 100 %, und 0 für das „unmögliche Ereignis", also eine Wahrscheinlichkeit von 0 %. Die Softmax-Funktion σ wandelt einen k-dimensionalen reellen Vektor in einen k-dimensionalen reellen Vektor um, dessen Komponenten im Intervall [0,1] liegen und sich zu 1 aufsummieren. Der Vorteil dieser Funktion ist, dass die Eingabewerte beliebige Werte aus $\mathbb{R}$ sein können, aber von der Softmax-Funktion auf das Intervall [0,1] mit der Komponentensumme 1 abgebildet werden, sodass sie als Wahrscheinlichkeiten interpretiert werden können. Wenn einer der Eingabewerte klein oder negativ (groß) ist, wandelt die Softmax-Funktion ihn in eine kleine (große) Wahrscheinlichkeit um. Die Softmax-Funktion findet häufig eine Anwendung beim maschinellen Lernen in einem Künstlichen Neuronalen Netz; vgl. PAAß/ HECKER (2020), S. 60-62.

In der nachfolgenden Abbildung123, die abermals einen „angepassten" Ausschnitt aus der Abbildung120 zeigt, wird die Berechnung des Vorhersagevektors $\widehat{\vec{u}^T}$ veranschaulicht.

Abbildung 123: Berechnung des Vorhersagevektors $\widehat{\vec{u}^T}$

Die Berechnung des Vorhersagevektors $\widehat{\vec{u}^T}$ erfolgt auf folgende Weise:

$$\widehat{\vec{u}^T} = \sigma\left(\vec{u_i}^T\right) = \left[\frac{e^{u_i}}{\sum_{j=1}^{R} e^{u_j}} \quad i = 1, \dots, R\right]$$

$$= \left[\frac{e^{u_1}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_2}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_3}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_4}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_5}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_6}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_7}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_8}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_9}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_{10}}}{\sum_{j=1}^{10} e^{u_j}}\right]$$

Dabei steht u_i für das i-te Element des Vektors $\vec{u}^T$. R ist die Anzahl der Elemente im Vektor $\vec{u}^T$ und beträgt hier $R = 10$. Das Symbol Σ repräsentiert die Summe über alle Elemente mit dem Laufindex j mit j=1,…,R.

Der Vorhersagevektor $\widehat{\vec{u}^T}$ weist folgende gerundete (Symbol „≈") Werte auf:

$$\widehat{\vec{u}^T} \approx [0.093 \quad 0.094 \quad 0.096 \quad 0.098 \quad 0.099 \quad 0.101 \quad 0.102 \quad 0.104 \quad 0.106 \quad 0.107]$$

Im nächsten Schritt werden die Fehlervektoren für die beiden Kontextwörter berechnet. Da sich die beiden Vektoren für die Kontextwörter $\overrightarrow{y_1}$ („die“) und $\overrightarrow{y_2}$ („berlin“) aus dem Trainingsmuster ergeben, werden die Fehlervektoren $\overrightarrow{m_1}$ und $\overrightarrow{m_2}$ berechnet, indem von den Vektoren $\overrightarrow{y_1}$ und $\overrightarrow{y_2}$ für die beiden Kontextwörter der Vorhersagevektor $\hat{\vec{u}}$ subtrahiert wird; vgl. GOLDBERG/ LEVY (2014), S. 8. Hier wird der Vorteil der Verwendung der Fenstergröße *m* deutlich. Sie reduziert die Komplexität der Berechnung, weil das Modell nur die Vorhersagefehler für die Kontextwörter innerhalb der vorgegebenen Fenstergröße *m* berechnen muss, für die hier $m = 1$ gilt.

Ein von Null verschiedener Fehlerwert im Fehlervektor deutet darauf hin, dass die Vorhersagen des Modells von den tatsächlichen Beobachtungen, hier also den beiden Vektoren $\overrightarrow{y_1}$ und $\overrightarrow{y_2}$ für die beiden Kontextwörter, abweichen. Der ideale Wert für den Fehlervektor wäre $\vec{0}$. Dies würde darauf hinweisen, dass das Modell „perfekte“ Vorhersagen trifft und keine Abweichung zwischen den Vorhersagen und den tatsächlichen Kontextwörtern vorhanden ist. In der Praxis ist es jedoch selten möglich, einen Fehlervektor von genau $\vec{0}$ zu erreichen, da nahezu immer Abweichungen existieren. Daher besteht das Hauptziel der *Optimierung* durch den Fehlervektor darin, diese Abweichung so weit wie möglich zu *minimieren*, um die Vorhersagegenauigkeit des gesamten Modells zu *maximieren*. Hiermit ist ein präzises und zugleich operationales Optimierungskriterium für die Berechnungen mithilfe der Word2Vec-Technik im betrachteten Künstlichen Neuronalen Netz definiert.

Aufgrund der unterschiedlichen Dimensionen zwischen dem Vorhersagevektor $\widehat{\vec{u}^T}$ (Dimension $1\ x\ 10$) und den Vektoren $\overrightarrow{y_1}$ und $\overrightarrow{y_2}$ (Dimensionen jeweils $10\ x\ 1$) wird der Vorhersagevektor transponiert. In der nachfolgenden Abbildung 124, die abermals einen „angepassten“ Ausschnitt aus der Abbildung120 zeigt, wird die Berechnung der Fehlervektoren $\overrightarrow{m_1}$ und $\overrightarrow{m_2}$ veranschaulicht.

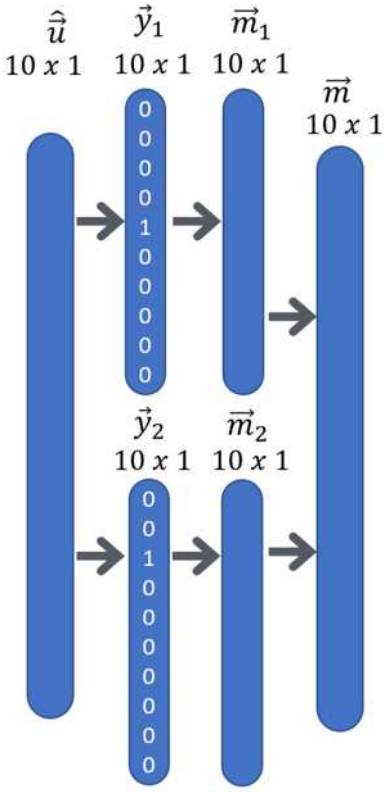


Abbildung 124: Berechnung der Fehlervektoren

Die Berechnung der Fehlervektoren $\overrightarrow{m_1}$ und $\overrightarrow{m_2}$ erfolgt auf folgende Weise:

$$\overrightarrow{m_1} = \hat{\vec{u}} - \overrightarrow{y_1}$$

$$\overrightarrow{m_1} = \begin{bmatrix} -0.907 \\ 0.094 \\ 0.096 \\ 0.098 \\ 0.099 \\ 0.101 \\ 0.102 \\ 0.104 \\ 0.106 \\ 0.107 \end{bmatrix}$$

$$\overrightarrow{m_2} = \hat{\vec{u}} - \overrightarrow{y_2}$$

$$\overrightarrow{m_2} = \begin{bmatrix} 0.093 \\ 0.094 \\ -0.904 \\ 0.098 \\ 0.099 \\ 0.101 \\ 0.102 \\ 0.104 \\ 0.106 \\ 0.107 \end{bmatrix}$$

Die Werte der Vorhersagefehler aus den Fehlervektoren $\overrightarrow{m_1}$ und $\overrightarrow{m_2}$ werden für beide Kontextwörter „die“ und „berlin“ aufsummiert und als (aggregierter) Fehlervektor $\vec{m}$ dargestellt:

$$\vec{m} = \sum_{c=1}^{C} \vec{m}_c$$

Hier steht *C* für die Anzahl der Fehlervektoren für Kontextwörter. In diesem Fall handelt es sich um die zwei Fehlervektoren $\overrightarrow{m_1}$ und $\overrightarrow{m_2}$ (mit $C = 2$), die addiert werden:

$$\vec{m} = \begin{bmatrix} -0.814 \\ 0.188 \\ -0.808 \\ 0.196 \\ 0.198 \\ 0.202 \\ 0.204 \\ 0.208 \\ 0.212 \\ 0.214 \end{bmatrix}$$

Nachdem der Fehlervektor $\vec{m}$ berechnet wurde, erfolgt die Aktualisierung der beiden Gewichtsmatrizen W_{Input} und W_{Output}, um das Modell zu „optimieren“. Dieser Prozess wird als „Backpropagation“ bezeichnet und basiert auf dem Gradientenabstiegsverfahren; vgl. RONG (2014), S. 17-20. Die Idee dahinter ist, die Werte in W_{Input} und W_{Output} schrittweise so anzupassen, dass der Fehlervektor $\vec{m}$ minimiert wird.

Zunächst erfolgt die Erläuterung der Aktualisierung der Gewichtsmatrix W_{Input}, die nachfolgend als $W_{Input_{neu}}$ bezeichnet wird.

Dazu wird der Fehlervektor $\vec{m}$ mit dem transponierten Vektor $\vec{h}$ aus dem Hidden-Layer multipliziert:

$$\vec{m} * \vec{h}^T$$

Die Berechnung der aktualisierten der Gewichtsmatrix $W_{Input_{neu}}$ erfolgt, indem das voranstehende Produkt mit der Lernrate η multipliziert wird und von der alten Gewichtsmatrix $W_{Input_{alt}}$ subtrahiert wird:

$$W_{Input_{neu}} = W_{Input_{alt}} - \eta * \vec{m} * \vec{h}^T$$

Die Lernrate η stellt abermals einen Hyperparameter dar. Sie steuert, wie schnell das Modell des Künstlichen Neuronalen Netzes an das jeweils betrachtete Vorhersageproblem angepasst wird. Die Lernrate bestimmt die Stärke der Gewichtsänderungen in den Gewichtsmatrizen W_{Output} und W_{Input}. Dieses Verfahren heißt Delta-Regel oder WIDROW-HOFF-Verfahren. Kleinere Lernraten erfordern mehr Trainingszyklen angesichts der kleineren Änderungen, die an den Gewichtungen bei jeder Aktualisierung vorgenommen werden, während größere Lernraten zu schnelleren Änderungen führen und weniger Trainingszyklen benötigen; vgl. MIRFENDRESKI (2022), S. 143. Ein Trainingszyklus umfasst eine Iteration über den gesamten Trainingsdatensatz. In der Regel bewegen sich die Lernraten zwischen 1 und 0 im Bereich des maschinellen Lernens; vgl. AICHELE (2021), S. 11; KLÜVER/KLÜVER (2021), S. 14; KLÜVER/KLÜVER/SCHMIDT (2021), S. 189. Für die Word2Vec-Technik wurde eine Lernrate zwischen 0.025 und 0.0001 genutzt, die von DI GENNARO/BUONANNO/PALMIERI (2021), S. 12328, als Ausgangspunkt für die Ermittlung einer optimalen Lernrate angesehen wird.

Die Berechnung mit den zuvor genannten Werten erfolgt, indem zunächst die Neuberechnung („Optimierung“) für die Gewichtsmatrix $W_{Input_{alt}}$ mithilfe der Lernrate $\eta = 0.025$ durchgeführt wird:

$$W_{Input_{neu}} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 * \begin{bmatrix} -0.814 \\ 0.188 \\ -0.808 \\ 0.196 \\ 0.198 \\ 0.202 \\ 0.204 \\ 0.208 \\ 0.212 \\ 0.214 \end{bmatrix} * [0.13 \quad 0.14 \quad 0.15]$$

Im nachfolgenden Schritt wird das Berechnungsergebnis von $\vec{m} * \vec{h}^T$ dargestellt:

$$W_{Input_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 * \begin{bmatrix} -0.10582 & -0.11396 & -0.1221 \\ 0.02444 & 0.02632 & 0.0282 \\ -0.10504 & -0.11312 & -0.1212 \\ 0.02548 & 0.02744 & 0.0294 \\ 0.02574 & 0.02772 & 0.0297 \\ 0.02626 & 0.02828 & 0.0303 \\ 0.02652 & 0.02856 & 0.0306 \\ 0.02704 & 0.02912 & 0.0312 \\ 0.02756 & 0.02968 & 0.0318 \\ 0.02782 & 0.02996 & 0.0321 \end{bmatrix}$$

Im nachfolgenden Schritt wird das Berechnungsergebnis hinsichtlich der Multiplikation mit der Lernrate dargestellt:

$$W_{Input_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix} - \begin{bmatrix} -0.00265 & -0.00285 & -0.00305 \\ 0.00061 & 0.00066 & 0.00071 \\ -0.00263 & -0.00283 & -0.00303 \\ 0.00064 & 0.00069 & 0.00074 \\ 0.00064 & 0.00069 & 0.00074 \\ 0.00066 & 0.00071 & 0.00076 \\ 0.00066 & 0.00071 & 0.00077 \\ 0.00068 & 0.00073 & 0.00078 \\ 0.00069 & 0.00074 & 0.0008 \\ 0.0007 & 0.00075 & 0.0008 \end{bmatrix}$$

Zuletzt erfolgt die Subtraktion der beiden o. a. Matrizen. Die Gewichtsmatrix $W_{Input_{neu}}$ sieht wie folgt aus (sie wird in den folgenden Iterationen weiter „optimiert“):

$$W_{Input_{neu}} \approx \begin{bmatrix} 0.1026 & 0.2028 & 0.3030 \\ 0.3993 & 0.4993 & 0.5992 \\ 0.7026 & 0.8028 & 0.9030 \\ 0.0993 & 0.1093 & 0.1193 \\ 0.1294 & 0.1393 & 0.1493 \\ 0.1593 & 0.1692 & 0.1792 \\ 0.1893 & 0.1992 & 0.2092 \\ 0.2193 & 0.2293 & 0.2392 \\ 0.2493 & 0.2592 & 0.2692 \\ 0.2793 & 0.2892 & 0.2992 \end{bmatrix}$$

Die Berechnung der Gewichtsmatrix $W_{Output_{neu}}$ erfolgt auf ähnliche Weise wie die Berechnung der Gewichtsmatrix $W_{Input_{neu}}$. Es ist jedoch zu beachten, dass die Gewichtsmatrix $W_{Output_{neu}}$, wie auch die Gewichtsmatrix $W_{Output_{alt}}$, eine $(3x10)$-Matrix ist. Daher müssen die transponierten Vektoren verwendet werden, um eine mathematisch zulässige Berechnung zu gewährleisten. Dies bedeutet, dass der Vektor $\vec{h}^T$ und der Fehlervektor $\vec{m}$ vor der Multiplikation transponiert werden müssen, um sicherzustellen, dass die Multiplikation beider Vektoren eine Matrix mit derselben Form wie $W_{Output_{alt}}$ bildet:

$$\vec{h} * \vec{m}^T$$

Die Formel für die Berechnung von $W_{Output_{Neu}}$ lautet:

$$W_{Output_{neu}} = W_{Output_{alt}} - \eta * \vec{h} * \vec{m}^T$$

Die Berechnung von $W_{Output_{neu}}$ mit den zuvor genannten Werten erfolgt folgendermaßen:

$$W_{Output_{neu}} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 * \begin{bmatrix} 0.13 \\ 0.14 \\ 0.15 \end{bmatrix} * \begin{bmatrix} -0.814 & 0.188 & -0.808 & 0.196 & 0.198 & 0.202 & 0.204 & 0.208 & 0.212 & 0.214 \end{bmatrix}$$

Im nachfolgenden Schritt wird das Berechnungsergebnis von $\vec{h} * \vec{m}^T$ dargestellt:

$$W_{Output_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 *$$

$$\begin{bmatrix} -0.10582 & 0.02444 & -0.10504 & 0.02548 & 0.02574 & 0.02626 & 0.02652 & 0.02704 & 0.02756 & 0.02782 \\ -0.11396 & 0.02632 & -0.11312 & 0.02744 & 0.02772 & 0.02828 & 0.02856 & 0.02912 & 0.02968 & 0.02996 \\ -0.1221 & 0.0282 & -0.1212 & 0.0294 & 0.0297 & 0.0303 & 0.0306 & 0.0312 & 0.0318 & 0.0321 \end{bmatrix}$$

Im nachfolgenden Schritt wird das Berechnungsergebnis der Multiplikation mit der Lernrate dargestellt:

$$W_{Output_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix} -$$

$$\begin{bmatrix} -0.00265 & 0.00061 & -0.00263 & 0.00064 & 0.00064 & 0.00066 & 0.00066 & 0.00068 & 0.00069 & 0.0007 \\ -0.00285 & 0.00066 & -0.00283 & 0.00069 & 0.00069 & 0.00071 & 0.00071 & 0.00073 & 0.00074 & 0.00075 \\ -0.00305 & 0.00071 & -0.00303 & 0.000743 & 0.00074 & 0.00076 & 0.00077 & 0.00078 & 0.0008 & 0.0008 \end{bmatrix}$$

Zuletzt erfolgt die Subtraktion der beiden voranstehenden Matrizen. Die Gewichtsmatrix $W_{Output_{neu}}$, die in folgenden Iterationen weiter „optimiert“ wird, sieht wie folgt aus:

$$W_{Output_{neu}} \approx$$

$$\begin{bmatrix} 0.10265 & 0.19939 & 0.30363 & 0.39936 & 0.49936 & 0.59934 & 0.69934 & 0.79932 & 0.89931 & 0.99305 \\ 0.11284 & 0.11934 & 0.13283 & 0.13931 & 0.14931 & 0.15929 & 0.16929 & 0.17927 & 0.18926 & 0.19925 \\ 0.21305 & 0.2193 & 0.23303 & 0.23927 & 0.24925 & 0.25924 & 0.26924 & 0.27922 & 0.28921 & 0.29920 \end{bmatrix}$$

Wenn die Aktualisierung der beiden Gewichtsmatrizen W_{Output} und W_{Input} abgeschlossen ist, wird die aktuelle Iteration beendet. Das Fenster der Größe m bewegt sich zum nächsten Zentrumswort (in diesem Fall das Wort „berlin“). Die nächste Iteration beginnt.

Zusammenfassend lässt sich sagen, dass das Skip-gram-Modell versucht, die Werte der Gewichtsmatrizen W_{Output} und W_{Input} mithilfe des Fehlervektors $\vec{m}$ so zu „optimieren“, dass die Kontextwörter bei gegebenem Zentrumswort mit maximaler Wahrscheinlichkeit vorhergesagt werden. Die Maximierung der Wahrscheinlichkeit der Vorhersage von Kontextwörtern führt zur Optimierung der Gewichtsmatrizen W_{Output} und W_{Input}. Im Wesentlichen werden mithilfe der Word2Vec-Technik auf diese Weise statistische – oftmals als „semantisch“ bezeichnete (darauf wird zurückgekommen) – Beziehungen zwischen Wörtern gelernt, indem sie Vektordarstellungen für Wörter aus einem zugrunde gelegten Textkorpus entwickelt und „optimiert“.

Die informationstechnische Implementierung des Ähnlichkeitsalgorithmus für ein ontologiegestütztes CBR-System in einer Cloud-Umgebung wird in diesem Beitrag nicht näher beschrieben, weil die Details vermutlich nur IT- und KI-Experten interessieren wird. Ausführliche Implementierungserläuterungen finden sich stattdessen in SETHUPATHY (2024), S. 470-535.

5 Kritische Reflexionen

5.1 Ontologiegestütztes Case-based-Reasoning für die Wiederverwendung von Erfahrungswissen über sicherheitskritische IT-Projekte

Die Konstruktion der sicherheitskritischen IT-Projekt-Ontologie hängt von einer Vielzahl von Konstruktionsentscheidungen ab. Obwohl nahezu jede Entscheidung im Rahmen der Konstruktion der sicherheitskritischen IT-Projekt-Ontologie als diskussionswürdig charakterisiert werden kann, fokussieren sich die nachfolgenden Ausführungen lediglich auf einige aus Sicht der Verfasser besonders prominente und diskussionswürdige Konstruktionsentscheidungen.

Zunächst kann kritisch angemerkt werden, dass eine noch unvollständige PM-Domain-Ontologie aus dem Projekt KI-LiveS zugrunde gelegt wurde. Die PM-Domänen-Ontologie verfügt weder über eine Versionierung noch über eine zitierfähige Dokumentation. Darüber hinaus konnte mithilfe von OWLReady2 festgestellt werden, dass die PM-Domänen-Ontologie fehlerhaft war, da Namensbezeichnungen doppelt vorlagen und somit eine Fehlermeldung ausgegeben wurde.

Die Konstruktion der sicherheitskritischen IT-Projekt-Ontologie beschränkte sich im Wesentlichen darauf, die PM-Domänen-Ontologie um die Aspekte von sicherheitskritischen IT-Projekten zu erweitern. Obwohl durch die Verwendung der PM-Domänen-Ontologie die Integration in das CBR-Tool jCORA vereinfacht wurde, gibt es eine Vielzahl von Klassen und Eigenschaften in der PM-Domänen-Ontologie, die für die sicherheitskritische IT-Projekt-Ontologie nicht relevant sind. Die vorgenommenen Restriktionen beziehen sich hauptsächlich auf die erweiterten Klassen, Relationen und Attribute, sodass kritisch angemerkt werden kann, dass für eine vollständig integre Anwendung der konstruierten Ontologie auch die bereits existierenden Entitäten der bestehenden PM-Domänen-Ontologie mit Restriktionen hätten versehen werden müssen. Diese Kritik bezieht sich auch auf die verwendete PRINCE2- und Risikomanagement-Ontologie. Darüber hinaus kann für die sicherheitskritische IT-Projekt-Ontologie nicht erwartet werden, dass alle sprachlichen Ausdrucksmittel zur Repräsentation des domänenspezifischen Wissens enthalten sind.

Die Wahl des Softwaretools Protégé sollte ebenfalls kritisch diskutiert werden. Die von OWLReady2 erkannten Fehler bezüglich doppelter Namensnennungen wurden von Protégé nicht erkannt. Auch bei der Konstruktion der Ontologie traten gelegentlich Fehler auf. Nennenswerte Fehler, die von den Verfassern bei der Konstruktion einer Ontologie mit Protégé gefunden wurden, sind folgende:

- Bei der Konstruktion komplexer Sachverhalte (z. B. Vererbung) kann ein Fehler in Protégé dazu führen, dass das Editor-Fenster von einer anderen Klasse aufgerufen wird, als zuvor vom Benutzer ausgewählt wurde.
- Bei der Installation von Plug-Ins kann es gelegentlich vorkommen, dass Plug-Ins nicht verwendet werden können und neu installiert werden müssen.

- Protégé kann wegen überlappender grafischer Elemente nicht mehr verwendet werden; dies tritt häufig beim Neustart von Protégé auf.
- Sowohl der HermiT als auch der Pellet Reasoner von Protégé haben die Fehler doppelter Namensnennungen nicht erkannt.

Der Einsatz des internetbasierten Ontologie-Editors WebProtégé könnte vermutlich die aufgezeigten Probleme mit Protégé beheben. Insbesondere der hohe Aktualisierungsgrad von WebProtégé könnte ein Indiz dafür sein, dass dieses eine deutlich kürzere Fehlerbehebungszeit hat als die Desktop-Version von Protégé. Bei einer Weiterentwicklung der sicherheitskritischen IT-Projekt-Ontologie könnte der Aufbau ausschließlich mit dem Ontologie-Editor WebProtégé erfolgen, um z. B. auch die Vorteile einer internetbasierten Anwendungssoftware zu nutzen. Der Benutzer muss sich jedoch darüber im Klaren sein, dass WebProtégé im Vergleich zu Desktop Protégé über einen geringeren Funktionsumfang verfügt.

Die Ontologie gewinnt durch die Kardinalitäten und SWRL-Regeln an Ausdrucksstärke. Allerdings kann das CBR-Tool jCORA keine SWRL-Regeln verarbeiten. Dennoch haben sich die Verfasser für die Konstruktion von SWRL-Regeln entschieden, um diese im Sinne einer eigenständigen Ontologie möglichst ausdrucksstark zu gestalten. Eine weitere Einschränkung der SWRL-Regeln ist die Verwendung der Regeln im betrieblichen Umfeld, was eine aufwendige Pflege erfordert.

Des Weiteren können die „subjektive“ Auswahl der Fälle sowie die geringe Anzahl von Fällen in der Fallbasis kritisiert werden. Die Auswahl der Fälle stellt einerseits sicher, dass die für die Modellierung der Fälle notwendigen sprachlichen Ausdrucksmittel durch die Ontologie zur Verfügung gestellt werden. Andererseits lässt sich darüber diskutieren, dass nicht überprüft wurde, ob sich die konstruierte Ontologie auch im Kontext weiterer denkbarer Fälle als praktikabel erweist. Die geringe Anzahl der Fälle in der Fallbasis lässt keine Aussage über die Laufzeit z. B. zur Berechnung von Ähnlichkeitswerten und zur Ermittlung ähnlichster Projekte zu. Zwar lässt sich erkennen, dass der Leistungsbedarf bei der Ähnlichkeitsberechnung der Fälle zunimmt. Es ist jedoch davon auszugehen, dass im betrieblichen Umfeld weit mehr Fälle in der Fallbasis vorhanden sind als in dem hier prototypisch betrachteten ontologiegestützten CBR-System, sodass die Laufzeit deutlich ansteigen kann. Eine Bewertung hinsichtlich der Laufzeit auf Basis einer realistischen Anzahl von Fällen in der Fallbasis wurde nicht durchgeführt. Es ist auch wichtig zu ermitteln, ab welcher Anzahl von Fällen das CBR-System „bessere“ Ergebnisse hinsichtlich der Ermittlung möglichst ähnlicher Fälle liefert.

Weiterhin lässt sich einwenden, dass aufgrund von Geheimhaltungsstufen und möglichen Sicherheitsbedenken nicht jede Klasse in der Ontologie sicherheitskritischer IT-Projekte transparent konstruiert werden konnte. Auch fehlt an einigen Stellen aufgrund möglicher Sicherheitsbedenken der konkrete Bezug zu praxisrelevanten Leistungsbeschreibungen. Zwar ist es möglich, die konkrete Ausprägung sicherheitskritischer Anforderungen über Instanzen abzuleiten, jedoch konnten weitergehende Sicherheitseigenschaften, die einer Geheimhaltungsstufe unterliegen, in der sicherheitskritischen IT-Projekt-Ontologie nicht ausreichend ausgedrückt werden.

Ein weiterer Kritikpunkt ist die Sammlung der Begriffe für die Klassenkonstruktion, die für diesen Beitrag durch eine informelle Abfrage über E-Mail erfolgte. Es wäre denkbar, dass weitere Begriffe genannt worden wären, die für die Konstruktion relevant gewesen wären, jedoch wegen der Wahl der Methode, einer formlosen E-Mail, übersehen wurden. Das informelle Verfahren wurde jedoch bewusst gewählt, um nicht ausschließlich auf das Feedback von Experten angewiesen zu sein, sondern auch auf einzelne Begriffe aus den Leistungsbeschreibungen zurückgreifen zu können, bei denen keine Sicherheitsbedenken bestehen. Darüber hinaus sollte durch die informelle Abfrage eine Einflussnahme auf die Beantwortung der Begriffe und Kompetenzfragen vermieden und gleichzeitig eine Dokumentation der Beantwortung erleichtert werden.

Darüber hinaus fehlt die Adaption der Lösung des ähnlichsten alten Projekts, das die geforderte Mindestähnlichkeit erfüllt, an die Beschreibung eines neuen Projekts für die Anwendung des ontologiegestützten Case-based Reasonings auf sicherheitskritische IT-Projekte, sodass streng genommen keine vollständige Anwendung des Case-based Reasonings vorliegt. Zwar wird konzeptionell erläutert, wie die Adaption erfolgen könnte (insbesondere der Erwerb von Adaptionswissen). Es fehlen jedoch konkrete Adaptionsregeln für die Domäne der sicherheitskritischen IT-Projekte sowie exemplarische Anwendungen dieser Adaptionsregeln auf sicherheitskritische IT-Projekte. Die Phasen Revise und Retain wurden ebenfalls nicht betrachtet. Dies bleibt späteren Forschungsarbeiten – u. a. vom zweiten Verfasser dieses Beitrags im Rahmen seiner Dissertation – vorbehalten.

Die sicherheitskritische IT-Projekt-Ontologie hätte zudem im betrieblichen Umfeld auf ihre Ausdrucksstärke getestet werden können. Die Rückmeldungen der Anwender im Rahmen von möglichen Tests hätten u. a. in die Verbesserung der Ontologie einfließen können. Denkbar wäre ein mögliches Vorgehen wie bei WEBER et al. (2023), S. 38-56, wo die Benutzeroberfläche des CBR-Tools jCORA durch einen Nutzertest in Form einer Umfrage evaluiert wurde.

Zuletzt kann die Auswahl des CBR-Tools jCORA kritisiert werden. Die Auswahl dieses CBR-Tools hätte aufgrund eines Anforderungskatalogs von potenziellen betrieblichen Nutzern des Case-based Reasonings erfolgen können. Dafür hätte sich beispielsweise der Analytic Hierarchy Process als betriebswirtschaftliche multikriterielle Bewertungstechnik angeboten, der beispielsweise von BEIßEL (2011), S. 49-132, insbesondere S. 85-132, sowohl für einen Ontologie-Editor als auch für ein CBR-Tool angewendet wurde.

5.2 Cloud-native-Anwendung für ein ontologiegestütztes Case-based-Reasoning-System

Die Konzeption eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung sowie die darauf aufbauende prototypische Implementierung einzelner Funktionen hängen von zahlreichen Design-Entscheidungen ab, die im Folgenden kritisch reflektiert werden.

Die erste Entscheidung betrifft die Wahl der Cloud-Umgebung. Die Microsoft Azure Cloud von Microsoft hätte sowohl als primäre als auch als sekundäre Plattform gewählt werden können. Bei einer Entwicklung in der Microsoft Azure Cloud müsste anstelle der primär verwendeten Entwicklungsumgebung Cloud9 die Entwicklungsumgebung Visual Studio Online von Microsoft verwendet werden. Insbesondere für die Entwicklung von Microsoft-nahen Anwendungen wird Visual Studio Online empfohlen. Die Google Cloud Platform, die als sekundäre Plattform verwendet wurde, hätte auch als primäre Plattform verwendet werden können. Insbesondere die browserbasierte Entwicklungsumgebung Google Colab ist für die Entwicklung mit Python für maschinelles Lernen, Big-Data-Analysen und alle KI-Algorithmen konzipiert. Colab bietet als eine Cloud-gehostete Version des Jupyter Notebook freien Zugriff auf Computerinfrastruktur wie Speicher, Arbeitsspeicher, Verarbeitungskapazität, Grafikprozessoren (GPUs) und Tensor-Verarbeitungseinheiten (TPUs) für KI-Algorithmen an. In diesem Beitrag wurde die Funktion `simStringBOS` basierend auf Word2Vec mithilfe von Google Colab entwickelt. Denkbar wäre, alle Funktionen über Colab in Form eines Jupyter Notebook zu implementieren und ausschließlich in der Cloud-Umgebung von Google zur Verfügung zu stellen.

Eine weitere Einschränkung ist der Verzicht auf eine Datenbank in der prototypischen Implementierung. Die Ontologie wird auf einem Fileserver in der Cloud-Umgebung abgelegt, um prototypisch die Machbarkeit eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung zu demonstrieren. Im betrieblichen Umfeld müsste die Ontologie jedoch in einer Datenbank gespeichert und durch Zugriffsrechte geschützt werden. In der prototypischen Implementierung dieses Beitrags ist dies nicht geschehen. Das gesamte Rollen- und Rechtemanagement wurde nicht ausreichend betrachtet. Durch das API-Gateway wird zwar ein zentraler Zugang geschaffen, dieser stellt jedoch nur einen Ansatzpunkt für ein Rollen- und Rechtemanagement dar. Weitere Komponenten, wie z. B. Authentifizierungsmechanismen, müssten in der Folge berücksichtigt werden. Grundsätzlich sollte die Manipulation einer Ontologie an eine Rolle, wie z. B. Administrator, gebunden sein.

Weitere Bedenken bestehen im Bereich des Datenschutzes. Dies gilt insbesondere, weil es sich bei sicherheitskritischen IT-Projekten um Projekte mit hochsensiblen Daten handelt, die in der Regel einer Klassifizierung unterliegen. Im Hinblick auf die Einhaltung des Datenschutzes sind bei der Konzeption eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung zwei Bereiche verbesserungswürdig.

Erstens ist die Auswahl des Cloud-Anbieters aus datenschutzrechtlicher Sicht ein in der Literatur kritisch diskutierter Punkt. Siehe beispielsweise KNEUPER (2021), S. 11; JÄGER/RIEKEN/ERNST (2020), S. 3-10; VOIGT/VON DEM BUSSCHE (2018), S. 315-317. Es muss sichergestellt werden, dass die deutschen Datenschutzbestimmungen eingehalten werden und kein unberechtigter Zugriff auf die Daten (z. B. durch andere ausländische Sicherheitsbehörden) möglich ist.

Zweitens sollte neben dem rechtlichen Datenschutz auch der technische Datenschutz in Form weiterer Funktionalitäten berücksichtigt werden. So wurden beispielsweise das bereits erwähnte Rechtemanagement sowie die Pseudonymisierung und Verschlüsselung personenbezogener Daten in dem hier vorgestellten Konzept noch nicht ausreichend berücksichtigt. Zwar

bieten sich durch die Verwendung des API-Gateways und durch die Implementierung mittels Serverless-Funktionen verschiedene Ansatzpunkte, um den technischen Datenschutz zu integrieren. Ein Ansatzpunkt könnte darin bestehen, die Kommunikation zwischen dem Client, dem API-Gateway und den Serverless-Funktionen durch sichere Übertragungsprotokolle wie HTTPS zu verschlüsseln. Aspekte dieser Art wurden in diesem Beitrag noch nicht berücksichtigt. Diese Einschränkung wurde bewusst in Kauf genommen, da der Schwerpunkt dieses Beitrags auf dem Nachweis der generellen Machbarkeit eines ontologiegestützten CBR-Systems für sicherheitskritische IT-Projekte lag.

Ein weiterer grundlegender Punkt, der neben dem Datenschutz eine wichtige Rolle spielt, ist die Sicherheit einer Cloud-Umgebung. Durch Hackerangriffe können in einer Cloud-Umgebung Daten entwendet werden, was bei sicherheitskritischen IT-Projekten mit einer Geheimhaltungsstufe eine ernsthafte Bedrohung darstellen kann. Darüber hinaus zeigen aktuelle Fälle, dass auch Cloud-Umgebungen nicht vor Angriffen geschützt sind und solche Angriffe dazu führen können, dass ganze Systeme durch einen Angriff gelöscht werden. Als Beispiel sei hier auf den aktuellen Fall bei CloudNordic verwiesen, bei dem ein Hackerangriff erheblichen Schaden verursachte, indem die gesamten Daten in der Cloud-Umgebung verschlüsselt wurden und dadurch unbrauchbar gemacht wurden; vgl. KUHN (2023).

Kritisch zu hinterfragen ist auch die geringe Anzahl der implementierten spezifischen Ähnlichkeitsfunktionen. Im Dictionary sind lediglich vier Ähnlichkeitstypen mit zugeordneten Ähnlichkeitsfunktionen vorhanden. Zudem sind die Ähnlichkeitstypen und die spezifischen Ähnlichkeitsfunktionen im Dictionary hartcodiert hinterlegt. Für den Einsatz im betrieblichen Umfeld bedeutet das Dictionary einen zusätzlichen Aufwand, da die Zuordnung und Pflege manuell erfolgen müssen. Eine zuverlässige automatische Ermittlung der Ähnlichkeitstypen wäre wünschenswert, wird aber in der praktischen Umsetzung kaum möglich sein.

Die spezifische Ähnlichkeitsfunktion `simStringBOS` stellt eine zentrale Ähnlichkeitsfunktion dar, die eine Ähnlichkeit zwischen zwei Wörtern aus dem Bereich sicherheitskritischer IT-Projekte mittels Word2Vec berechnet. Es ist von entscheidender Bedeutung, dass bei der Implementierung dieser Funktion verschiedene kritische Aspekte berücksichtigt werden, die sorgfältig bewertet werden müssen. Zunächst ist die grundsätzlich diskutable Auslegung der Ähnlichkeit von Word2Vec zu kritisieren. Im mathematischen Sinne wird Ähnlichkeit in der Regel als Maß für die Nähe oder Korrelation zwischen zwei Objekten definiert. Bei Word2Vec wird die Ähnlichkeit jedoch anhand der räumlichen Nähe der Vektoren im Vektorraum berechnet. Wörter, die in ähnlichen Kontexten vorkommen und ähnliche Bedeutungen haben, liegen im Vektorraum nahe beieinander. Es wird davon ausgegangen, dass Wörter, die in ähnlichen Kontexten vorkommen, auch ähnliche Bedeutungen haben. Daher wird die räumliche Nähe der Vektoren als Maß für die Ähnlichkeit zwischen den Wörtern verwendet. Es ist wichtig zu beachten, dass die mit Word2Vec berechnete Ähnlichkeit auf Mustern (Zentrumswörter und Kontextwörter) basiert, die das Modell aus dem Textkorpus gelernt hat. Es kann Situationen geben, in denen die Ähnlichkeit nicht immer der intuitiven Bedeutung oder dem menschlichen Ver-

ständnis von Ähnlichkeit entspricht. Die Ergebnisse hängen stark von der Qualität und Repräsentativität des Textkorpus ab. Es ist zusätzlich anzumerken, dass Word2Vec als statistische Methode auf Wahrscheinlichkeiten und Verteilungen basiert. Infolgedessen kann streng genommen der Begriff „Ähnlichkeit“ nicht immer in dem klassischen mathematischen Sinne interpretiert werden, sondern es ist eher angemessen, den Begriff „Wahrscheinlichkeit“ zu verwenden.

Neben der diskutablen Auslegung der Ähnlichkeit von Word2Vec sind bei der Implementierung der Ähnlichkeitsfunktion `simStringBOS` ebenso einige Entscheidungen zu hinterfragen. Zunächst lässt sich über die Auswahl und die vorgenommene Anonymisierung der Leistungsbeschreibungen diskutieren. Zwar werden drei Leistungsbeschreibungen als Trainingsgrundlage verwendet, die zusammen 1.300 Seiten umfassen, jedoch könnte diese Trainingsgrundlage um weitere Leistungsbeschreibungen erweitert werden, um eine breitere Basis zu schaffen. Wie bereits in den vorherigen Erläuterungen dargestellt, besteht die Herausforderung darin, einerseits praxisrelevante Trainingsgrundlagen zur Verfügung zu stellen, andererseits aber auch kritische Informationen oder Dokumente mit Geheimhaltungsstufen aus den Trainingsgrundlagen herauszunehmen. Um dieser Herausforderung gerecht zu werden, wurden die Anonymisierung und die Auswahl auf drei Leistungsbeschreibungen beschränkt.

Die Datenaufbereitung des Textdokuments in der Funktion `simStringBOS` kann in der Vorverarbeitung ebenfalls diskutiert werden. Insbesondere werden Kapitelnummern und Sonderzeichen wie „(“ nicht entfernt. Eine automatisierte Entfernung ist zwar möglich, würde aber dazu führen, dass Zahlen wie „110“ und „112“, die im Bereich sicherheitskritischer IT-Projekte eine besondere Bedeutung haben, ebenfalls entfernt würden. Denkbar wären die Definition von Ausnahmen und eine manuelle Bearbeitung des Textdokuments. Für die Darstellung der Machbarkeit reicht die durchgeführte Datenaufbereitung zwar aus, die korrekte Berechnungsergebnisse für die Ähnlichkeit zweier Wörter liefert. Insgesamt ist jedoch festzuhalten, dass die Datenaufbereitung durch weitere Datenbereinigungen verbessert werden kann. Dabei ist abzuwägen, inwieweit die Textgrundlage durch eine zu starke Datenaufbereitung manipuliert und der ursprüngliche Inhalt zu stark verändert wird.

Es gibt auch neue Technologien wie den Transformer-Ansatz (Transformer-based Language Models sowie Large Language Models), um die Ähnlichkeit von Wörtern zu bestimmen. Ein Transformer ist ein Typ Künstlicher Neuronaler Netze, der erstmals 2017 von VASWANI et al. vorgestellt wurde. Der wesentliche Kern eines Transformers ist seine Fähigkeit, Informationen (z. B. aus natürlichsprachlichen Texten) parallel zu verarbeiten und damit das zugrunde liegende Künstliche Neuronale Netz schnell zu trainieren; vgl. VASWANI et al. (2017), S. 2. Der Algorithmus verwendet Multi-Head-Attention-Mechanismen, um die Bedeutung von jedem Wort im Kontext seiner Umgebung zu verstehen; vgl. VASWANI et al. (2017), S. 1. Obwohl sowohl Word2Vec als auch Transformer auf Künstlichen Neuronalen Netzen basieren, gibt es einige wichtige Unterschiede. Word2Vec beruht auf einem relativ einfachen Künstlichen Neuronalen Netz, das Worteinbettungen lernt. Ein Transformer hingegen besitzt eine komplexere,

mehrschichtige Architektur für Künstliche Neuronale Netze, die auf dem „Attention-Mechanismus“ basiert. Der Attention-Mechanismus (englisch: attentional mechanism) ist ein Konzept des maschinellen Lernens, welches von LUONG/PHAM/MANNING publiziert wurde. Es dient dazu, bestimmte Teile eines Inputs stärker zu berücksichtigen als andere, indem eine Gewichtung für jedes Element berechnet wird; vgl. LUONG/PHAM/MANNING (2015), S. 1. Bei der Verarbeitung von natürlichsprachlichen Texten wird der Attention-Mechanismus im Rahmen des Transformers eingesetzt, um die Bedeutung (Häufigkeit von Koinzidenzen) eines Worts im Kontext seiner Umgebung zu verstehen; vgl. LUONG/PHAM/MANNING (2015), S. 3-4. Hierbei berechnet der Transformer eine Gewichtung für jedes Wort im Text und berücksichtigt dann bei der Verarbeitung des Textes nur die Worte mit den höchsten Gewichtungen. Dadurch kann der Transformer die Bedeutung (Häufigkeit von Koinzidenzen) eines Wortes im Kontext seiner Umgebung verstehen; vgl. LUONG/PHAM/MANNING (2015), S. 8. Word2Vec verarbeitet dagegen Texte *sequenziell* und berücksichtigt nur Wortbeziehungen in einem begrenzten Kontext (Fenstergröße), während der Transformer eine *parallele* Verarbeitung ermöglicht und die Bedeutung jedes Wortes im Kontext seiner Umgebung versteht. Der Transformer-Ansatz hat sich bei der Verarbeitung natürlichsprachlicher Texte als sehr erfolgreich erwiesen. Ein weithin bekanntes Beispiel ist ChatGPT. Die Leistung des Transformer-Ansatzes ist höher als bei Word2Vec, da die Methode eine parallele Verarbeitung ermöglicht.

Zusammenfassend bietet der Transformer-Ansatz im Vergleich zu Word2Vec eine komplexere Modellarchitektur für Künstliche Neuronale Netze, was zu einem schnelleren Training und besseren Ergebnissen bei der Berechnung von Wortähnlichkeiten führen kann und deshalb eine interessante Option für den Einsatz in einem CBR-System darstellt. In der aktuellen Literatur wird die Verwendung eines Transformers als vielversprechend angesehen. Siehe beispielsweise NIGHOJKAR/KHLYZOVA/LICATO (2022), S. 6; TABINDA KOKAB/ASGHAR/NAZ (2022), S. 11; MOHD/DHASMANA/UPADHYAY (2021), S. 2399. In den genannten Quellen wird auch ein Vergleich zwischen einem Transformer und einem Word2Vec-Ansatz vorgenommen.

Im Rahmen der Implementierung der Funktion `simStringBOS` ist die Auswahl der Parameterwerte kritisch zu betrachten. Obwohl in diesem Beitrag verschiedene Werte für die Parameter ausgewählt und getestet wurden, lassen sich weitere Parameterkombinationen vorstellen, die über den Rahmen dieses Beitrags hinausgehen.

Grundsätzlich können aufgrund des prototypischen Charakters die implementierten Funktionen in der Hinsicht kritisiert werden, dass folgende Punkte der Softwareentwicklung nicht ausreichend berücksichtigt wurden:

- Es wurde keine Fehlerbehandlung vorgesehen, um eventuelle Fehleingaben eines Benutzers abzufangen. Dies kann beispielsweise der Fall sein, wenn eine Funktion für die Ähnlichkeitsberechnung einen String-Datentyp erwartet, jedoch ein Integer-Datentyp vom Benutzer eingegeben wird.
- Der Quellcode wird nicht in ein Software-Repository geladen und versioniert.
- Es wurden keine automatisierten Softwaretests durchgeführt.

Auch wenn in diesem Beitrag die Gestaltung des User Interfaces als Klick-Prototyp auf der Publikation WEBER et al. (2023) basiert, lässt sich über die dort erfolgten Designentscheidungen diskutieren. Für das Design des User Interfaces wurde der Usability Engineering Lifecyle nach NIELSEN gewählt. Dies kann aus zwei Perspektiven kritisiert werden. Zunächst ist der Usability Engineering Lifecycle nach MAYHEW detaillierter als der Usability Engineering Lifecycle nach NIELSEN. Zweitens hätten die Aspekte von NIELSEN mit denen von MAYHEW kombiniert werden können. Ein weiterer Kritikpunkt im Rahmen des Designs des User Interfaces ist, dass bei den Designentscheidungen die Barrierefreiheit des Klick-Prototyps zu wenig Berücksichtigung gefunden hat. Die Barrierefreiheit gewinnt für die Anwendungen im betrieblichen Umfeld eine zunehmende Bedeutung. Dies zeigt sich sowohl in der Barrierefreie-Informationstechnik-Verordnung (BITV) als auch in der ISO-Norm 9241 (Ergonomie der Mensch-System-Interaktion). Eine weitere Einschränkung des User Interfaces ist, dass der Klick-Prototyp lediglich in Form des Webdesigns gestaltet wurde und mobile Anwendungen nicht beachtet wurden. Für weitere kritisch reflektierte Designentscheidungen, auf die hier nicht weiter eingegangen wird, wird auf WEBER et al. (2023), S. 103-104, verwiesen.

Die Auswahl des Ähnlichkeitsalgorithmus für die beispielhafte Implementierung der Konzeption eines ontologiegestützten CBR-Systems als Cloud-native-Anwendung kann kritisch betrachtet werden. Die Auswahl von Anwendungskomponenten beruht auf subjektiven Kriterien, wie z. B. hinsichtlich der zu erwartenden Komplexität der Implementierung, der notwendigen automatischen Verarbeitung einer Ontologie und des möglichen hohen Ressourcenverbrauchs bei der Ähnlichkeitsberechnung. Außerdem ist nicht auszuschließen, dass bei einer vollständigen Implementierung als Cloud-native-Anwendung Probleme an anderer Stelle auftreten könnten, beispielsweise bei der Konfiguration der RESTful-API-Struktur für die Interaktion zwischen Frontend und Backend oder bei der Bereitstellung der Benutzeroberfläche in einer Cloud-Umgebung. Dieser Beitrag offeriert streng genommen kein vollständiges Konzept, das alle Bereiche (Frontend und Backend) im Zusammenspiel betrachtet, Stattdessen werden die einzelnen Bereiche getrennt voneinander betrachtet und der Schwerpunkt liegt auf der Implementierung des Ähnlichkeitsalgorithmus.

Schließlich muss die Cloud-native-Anwendung insgesamt kritisch bewertet werden. Ein Fehler in einer Funktion bleibt nicht isoliert, sondern kann sich auf den gesamten Ähnlichkeitsalgorithmus auswirken. Je mehr Funktionen implementiert werden und miteinander interagieren, desto wichtiger wird das Testen und Überwachen der Funktionen. Zwar wird die Komplexität der Funktionen durch isolierte Funktionen reduziert, aber das entstehende verteilte System über Cloud-Anbieter hinweg erhöht die Komplexität des Backends. Statt eines Monolithen müssen mehrere Funktionen parallel überwacht werden.

6 Fazit zu den gewonnenen wissenschaftlichen Erkenntnissen

Ausgangspunkt dieses Beitrags waren die eingangs vorgestellten wissenschaftlichen Probleme, die mittels der folgenden intendierten wissenschaftlichen Ergebnisse gelöst werden sollten:

- sicherheitskritische IT-Projekt-Ontologie,
- CBR-System mit integrierter Ontologie für sicherheitskritische IT-Projekte,
- sicherheitskritische IT-Projekte als Fälle in einem CBR-System,
- Ähnlichkeitsberechnung im CBR-System,
- Ähnlichkeitsalgorithmus als Serverless-Funktionen in einer Cloud-Umgebung sowie
- Ähnlichkeitsfunktionen als Serverless-Funktionen in einer Cloud-Umgebung, einschließlich spezifischer Ähnlichkeitsfunktionen zur Verarbeitung qualitativer Informationen aus sicherheitskritischen IT-Projekten unter Verwendung Künstlicher Neuronaler Netze.

Nachfolgend wird ein Fazit zu den oben genannten, angestrebten wissenschaftlichen Ergebnissen gezogen, um zu überprüfen, ob dieser Beitrag tatsächlich zu den intendierten Ergebnissen geführt hat.

Für die Konstruktion einer sicherheitskritischen IT-Projekt-Ontologie wurde die Vorgehensweise in Anlehnung an NOY/MCGUINNESS entwickelt, die um eine zusätzliche Aktivität – Definition von Regeln – erweitert wurde. Um eine praxisorientierte sicherheitskritische IT-Projekt-Ontologie zu konstruieren, wurden Kompetenzfragen und Begriffe von Experten formuliert und Begriffe aus relevanten Leistungsbeschreibungen unter Beachtung der Geheimhaltungsklassifizierung verwendet. Die sicherheitskritische IT-Projekt-Ontologie wurde mit dem Ontologie-Editor Protégé konstruiert. Hierbei wurden die PM-Domänen-Ontologie sowie die PRINCE2- und Risikomanagement-Ontologie als Grundlagen verwendet. Das intendierte wissenschaftliche Ergebnis einer sicherheitskritischen IT-Projekt-Ontologie kann unter Berücksichtigung der in Kapitel 5.1 beschriebenen Einschränkungen als erreicht betrachtet werden.

Als CBR-Tool für die Konstruktion ontologiegestützter CBR-Systeme wurde jCORA verwendet, das heterogene Fallbasen unterstützt und über einen integrierten Ähnlichkeitsalgorithmus verfügt. Die Implementierung der sicherheitskritischen IT-Projekt-Ontologie in ein derartiges CBR-System stellte das zweite intendierte wissenschaftliche Ergebnis dar. Es kann als erfüllt angesehen werden, weil eine erfolgreiche Integration der Ontologie in das CBR-Tool jCORA zur Fallkonstruktion und Ähnlichkeitsberechnung stattgefunden hat.

Um das intendierte wissenschaftliche Ergebnis, sicherheitskritische IT-Projekte als Fälle in einem ontologiegestützten CBR-System zu erfassen, wurden solche Fälle auf der Basis der sicherheitskritischen IT-Projekt-Ontologie spezifiziert. Das intendierte wissenschaftliche Ergebnis kann als erfüllt angesehen werden, da in exemplarischer Weise drei Fälle für sicherheitskritische IT-Projekte spezifiziert wurden.

Eine Ähnlichkeitsberechnung mit den spezifizierten Fällen wurde im ontologiegestützten CBR-System durchgeführt. Aber es kann nicht mit Sicherheit gesagt werden, ob diese Berechnung korrekt durchgeführt wurde. Es wurden Einschränkungen festgestellt, wie z. B. mangelnde Anwendungsperformance und fehlerhafte Ähnlichkeitsberechnungen bei der Verwendung globaler Instanzen. Kritisch anzumerken ist auch, dass keine Gewichtung der Instanzeigenschaften vorgenommen wurde, sodass alle Instanzeigenschaften der Fälle einheitlich mit dem Gewicht 1,0 gewichtet wurden, obwohl in der Praxis unterschiedliche Gewichtungen erforderlich sein können. Diese Vorgehensweise wurde jedoch gewählt, um lediglich die grundsätzliche Machbarkeit des hier vorgestellten Konzepts für ontologiegestützte CBR-Systeme zu demonstrieren. Außerdem fehlen spezifische Ähnlichkeitsfunktionen, sodass die universellen Ähnlichkeitsfunktionen unzureichend sein und zu fehlerhaften Berechnungen führen können. Das intendierte wissenschaftliche Ergebnis der Ähnlichkeitsberechnung im CBR-System wird daher nur als teilweise erreicht angesehen.

Um das intendierte wissenschaftliche Ergebnis zu erreichen, den Ähnlichkeitsalgorithmus als Serverless-Funktion bereitzustellen, um die Machbarkeit für ontologiegestütztes Case-based Reasoning als Cloud-native-Anwendung zu demonstrieren, wurden Python als Programmiersprache sowie Amazon Web Services (AWS) und die Google Cloud Platform (GCP) als Cloud-Umgebungen ausgewählt. Für die Verarbeitung der Ontologien wurde das Python-Modul OWLReady2 eingesetzt. Beim initialen Einlesen der Ontologien mit OWLReady2 in die Cloud-Umgebung wurde festgestellt, dass die zugrunde gelegte PM-Domain-Ontologie Inkonsistenzen aufwies, die ein Einlesen unmöglich machten. Der Ontologie-Editor Protégé hatte diese Inkonsistenzen nicht gemeldet. Nach manueller Behebung der Inkonsistenzen in der Ontologie (doppelte Namensnennungen) konnte ein Import mit OWLReady2 durchgeführt werden. Sowohl OWLReady2 als auch Protégé verwenden HermiT als Reasoner, aber Protégé zeigte die Inkonsistenzen nicht an.

Eine weitere Herausforderung war die Implementierung des Ähnlichkeitsalgorithmus, der als Serverless-Funktion implementiert wurde. Es wurde festgestellt, dass die berechnete Ähnlichkeit zwischen zwei Instanzen trotz korrekter Anwendung des Algorithmus als zu gering empfunden wurde. Um diesem Eindruck entgegenzuwirken, wurden nicht nur gemeinsame Klasseneigenschaften, sondern auch Klasseneigenschaften des gleichen Ähnlichkeitstyps zum Vergleich herangezogen. Obwohl der Algorithmus in einigen Bereichen, insbesondere durch die Verwendung eines Dictionarys, hartcodiert wurde, zeigt die Implementierung grundsätzlich, dass der Ähnlichkeitsalgorithmus durch Serverless-Funktionen bereitgestellt werden kann und auch Ansatzpunkte für eine Weiterentwicklung bietet. Da der vollständige Ähnlichkeitsalgorithmus als Serverless-Funktion implementiert wurde, die zu einer korrekt berechneten Ähnlichkeit geführt hat, ist das intendierte wissenschaftliche Ergebnis als erreicht anzusehen.

Das letzte intendierte wissenschaftliche Ergebnis dieses Beitrags ist die Implementierung spezifischer Ähnlichkeitsfunktionen für die Verarbeitung qualitativer Informationen aus sicherheitskritischen IT-Projekten mit Künstlichen Neuronalen Netzen. Es wurden spezifische Ähnlichkeitsfunktionen basierend auf Ähnlichkeitstabellen (beispielhaft in der Funktion `simTCV`

codiert), Datentypen wie Boolean sowie zwei spezifische Ähnlichkeitsfunktionen basierend auf Word2Vec-Modellen implementiert. Zu Demonstrationszwecken wurde ein Textkorpus aus Leistungsbeschreibungen für sicherheitskritische IT-Projekte aufbereitet und in zwei Modellen trainiert. Die spezifische Ähnlichkeitsfunktion ist in der Lage, die Ähnlichkeit zweier Wörter aus den Leistungsbeschreibungen zu berechnen. Die Herausforderung bei der Berechnung des Modells liegt in der Parametrisierung. Vordefinierte Modelle können hier Abhilfe schaffen, sind aber domänenunabhängig. Auch wenn aktuelle Entwicklungen wie ChatGPT zeigen, dass Word2Vec-Modelle in Zukunft von Transformer-Ansätzen abgelöst werden könnten, gibt es derzeit keine Implementierungen, um solche Algorithmen für ontologiegestütztes Case-based Reasoning zu verwenden. Da die spezifischen Ähnlichkeitsfunktionen als Serverless-Funktionen implementiert wurden, kann das intendierte wissenschaftliche Ergebnis als erreicht angesehen werden.

Zusammenfassend werden die intendierten wissenschaftlichen Ergebnisse mit den tatsächlich erreichten wissenschaftlichen Ergebnissen mithilfe von „Harvey-Bällen“:

Harvey-Ball	**Beschreibung**
	kein Ergebnis
	Ergebnis nur in geringen Teilen erbracht
	Ergebnis teilweise erbracht
	Ergebnis zu großen Teilen erbracht
	Ergebnis vollständig erbracht

Tabelle 62: Harvey-Ball-Definitionen
für den qualitativen Vergleich der intendierten wissenschaftlichen Ergebnisse

für den Grad der Zielerreichung miteinander verglichen. Das Vergleichsergebnis wird in der nachfolgenden Tabelle 63 dargestellt. Zu ausführlicheren Begründungen vgl. SETHUPATHY (2024), S. 552-553.

intendiertes wissenschaftliches Ergebnis	**tatsächlich erreichtes wissenschaftliches Ergebnis**	**Erfüllungs-grad**
sicherheitskritische IT-Projekt-Ontologie	sicherheitskritische IT-Projekt- Ontologie	●
CBR-System mit integrierter Ontologie für sicherheitskritische IT-Projekte	CBR-System mit integrierter Ontologie für sicherheitskritische IT-Projekte	●
sicherheitskritische IT-Projekte in Form von Fällen in einem CBR-System	sicherheitskritische IT-Projekte in Form von Fällen in einem CBR-System	●
Ähnlichkeitsberechnung im CBR-System	Ähnlichkeitsberechnung im CBR-System mit universellen Ähnlichkeitsfunktionen	◑
Ähnlichkeitsalgorithmus als Serverless-Funktionen auf einer Cloud-Umgebung	Ähnlichkeitsalgorithmus als Serverless-Funktionen auf einer Cloud-Umgebung	●
Ähnlichkeitsfunktionen als Server-less-Funktionen in einer Cloud-Umgebung einschließlich zwei spezifischer Ähnlichkeitsfunktionen zur Verarbeitung von qualitativen Informationen aus sicherheits-kritischen IT-Projekten unter der Verwendung von Künstlichen Neuronalen Netzen	Ähnlichkeitsfunktionen als Server-less-Funktionen in einer Cloud-Umgebung einschließlich zwei spezifischer Ähnlichkeitsfunktio-nen zur Verarbeitung von qualitati-ven Informationen aus sicherheits-kritischen IT-Projekten unter der Verwendung von Künstlichen Neuronalen Netzen	●

Tabelle 63: Vergleich zwischen intendierten
und tatsächlich erreichten wissenschaftlichen Ergebnissen

7 Ausblick auf weiteren Forschungsbedarf

Die in diesem Beitrag bereits angedeuteten Weiterentwicklungspotenziale bieten eine Vielzahl interessanter noch zu untersuchender Fragestellungen.

Der erste Ansatzpunkt für vertiefte Untersuchungen ist die Weiterentwicklung der sicherheitskritischen IT-Projekt-Ontologie um zusätzliche sprachliche Ausdrucksmittel und domänenspezifische Regeln. Die Konstruktion weiterer SWRL-Regeln würde die Aussagekraft der sicherheitskritischen IT-Projekt-Ontologie ausbauen und könnte auch als Grundlage für die Entwicklung spezifischer Adaptionsregeln dienen. Es wäre denkbar, weitere sprachliche Ausdrucksmittel durch die Einbeziehung zusätzlicher Experten sowie weiterer Leistungsbeschreibungen zu gewinnen und sprachliche Ausdrucksmittel aus anderen Projektmanagementmethoden wie PMI oder Scrum bereitzustellen und in der sicherheitskritischen IT-Projekt-Ontologie zu integrieren. Beispielsweise wäre es wünschenswert, Adaptionsregeln zu implementieren, die einen Austausch zwischen Projektmanagementmethoden ermöglichen, sodass ein Projekt, das mit PRINCE2 durchgeführt wurde, durch eine Adaptionsregel Vorschläge für neue Projekte hinsichtlich einer Anpassung an Projektmanagementmethoden wie PMI oder Scrum erhält.

Die Eingabe von Projektwissen und die Pflege der Fallbasis könnten durch Schnittstellen zu betrieblichen (Software-)Anwendungen (z. B. Sharepoint und SAP-Systeme) automatisiert werden. Die Schnittstellen zu den betrieblichen Anwendungen könnten als eigenständige Serverless-Funktionen in einer Cloud-Umgebung implementiert werden, die Daten aus den betrieblichen Anwendungen abrufen und in einem ontologiegestützten CBR-System für eine Cloud-native-Anwendung anzeigen. In einer weiteren Untersuchung könnte geprüft werden, inwieweit eine Automatisierung der Eingabe von Projektwissen möglich ist oder zumindest in einem ersten Schritt eine Vorverarbeitung des Projektwissens automatisiert erfolgen könnte. Eine Beschleunigung der Fallerfassung könnte auch durch die Generierung von Templates als Vorlagen ermöglicht werden. Die Mustererkennung für Fall-Templates könnte mithilfe von KI-Techniken erfolgen. Denkbar wäre die Verwendung von Doc2Vec, das im weiteren Verlauf des Ausblicks beschrieben wird, um Muster in einer großen Anzahl von Dokumenten zu erkennen und für die Erstellung von Fall-Templates zu verwenden.

Das CBR-Tool jCORA stellt einen anderen Ansatzpunkt für Weiterentwicklungen dar, die darauf abzielen, die derzeit vorhandenen Einschränkungen zu überwinden und das CBR-Tool auf eine „zukunftsfähige“ Informationstechnik vorzubereiten. In diesem Beitrag wurden exemplarisch das Einlesen und Verarbeiten einer Ontologie, der Ähnlichkeitsalgorithmus mit spezifischen Ähnlichkeitsfunktionen und ein User Interface als Klick-Prototyp vorgestellt, die für den Einsatz als eine Cloud-native-Anwendung ausgelegt sind. In einer vertieften Untersuchung wäre es wünschenswert, alle isoliert betrachteten Bereiche als eine voll funktionsfähige Cloud-native-Anwendung bereitzustellen und um zusätzliche spezifische Ähnlichkeitsfunktionen zu erweitern.

Im Rahmen einer Weiterentwicklung als Cloud-native-Anwendung ist es darüber hinaus denkbar, durch den Einsatz des Python-Moduls OWLReady2 das ontologiegestützte Case-based Reasoning um Ontologie-Editierfunktionen zu erweitern, um beispielsweise bei notwendigen Ergänzungen einer Ontologie nicht immer auf den Ontologie-Editor Protégé zurückgreifen zu müssen.

Ein anderer Ansatzpunkt für weitere Untersuchungen ist der Word2Vec-Algorithmus, der für die zwei spezifischen Ähnlichkeitsfunktionen `simStringBOS` und `simPreTrained` verwendet wurde. Hier sind verschiedene Weiterentwicklungen möglich. Zum einen kann der Word2Vec-Algorithmus mit anderen Python-Modulen (z. B. Tensorflow) hinsichtlich des Trainings verbessert werden, sodass das Künstliche Neuronale Netz um Netzebenen erweitert werden kann. Darüber hinaus wäre es wünschenswert, neben einem Word2Vec-Algorithmus auch einen Transformer-Algorithmus zu verwenden und diesen als spezifische Ähnlichkeitsfunktion zur Verfügung zu stellen. Obwohl es derzeit keine Module gibt, die ein ähnliches Ergebnis wie ChatGPT oder Bidirectional Encoder Representations from Transformers (BERT) liefern, kann mit Tensorflow eine grundlegende Anwendung mit dem Transformer-Ansatz in einem Python-Projekt erstellt werden. Eine Dokumentation, wie ein allgemeiner Transformer-Ansatz in einem Python-Projekt eingesetzt werden kann, findet sich in TENSORFLOW (2023). Diese Dokumentation beschreibt beispielhaft die Verwendung des Transformer-Ansatzes für die Übersetzung natürlichsprachlicher Texte. Eine inhaltliche Ähnlichkeitsbetrachtung, z. B. zwischen Sätzen oder Wörtern, ist ebenfalls möglich, müsste aber in einer weiterführenden Untersuchung evaluiert werden. Aktuelle Publikationen zeigen diese Möglichkeiten auf; vgl. beispielsweise BAER/PURVES (2023), S. 56; WORTH (2023), S. 14; ZHANG et al. (2023), S. 3. In den genannten Quellen wird die grundsätzliche Eignung des Transformer-Ansatzes für die Ähnlichkeitsprüfung natürlichsprachlicher Texte beschrieben.

In einer vertieften Untersuchung wäre es wünschenswert, diese Ansätze als eigenständige spezifische Ähnlichkeitsfunktionen als Serverless-Funktionen zu implementieren und diese auch in einem ontologiegestützten CBR-System für eine Cloud-native-Anwendung zu integrieren. Dazu könnten folgende Forschungsfragen für Folgeuntersuchungen formuliert werden:

- Wie kann der Transformer-Ansatz in ein ontologiegestütztes CBR-System integriert werden?
- Wie können Ontologien in Kombination mit Transformer-Algorithmen genutzt werden, um spezifische Adaptionsregeln zu entwickeln?

In diesem Beitrag wurden nur zwei spezifische Ähnlichkeitsfunktionen verwendet, die auf der Word2Vec-Technik basieren (`simStringBOS` und `simPreTrained`). In weiterführenden Untersuchungen könnten zusätzliche spezifische Ähnlichkeitsfunktionen implementiert werden, die mit unterschiedlichen Dokumenten trainiert werden. Basierend auf dem Berechnungsmodell Word2Vec existiert in Gensim das Submodul Doc2Vec, das einen vielversprechenden An-

satz zum Vergleich von Dokumenten bietet. Doc2Vec ermöglicht die Ermittlung der inhaltlichen Ähnlichkeit zwischen Dokumenten. Folgende Forschungsfragen können mit Doc2Vec beantwortet werden:

- Welche zwei Dokumente aus einer Menge von Dokumenten sind einander am ähnlichsten?
- Können Absätze, Passagen oder einzelne Sätze gefunden werden, die anderen Absätzen, Passagen oder einzelnen Sätzen desselben Dokuments oder anderer Dokumente ähneln?

Des Weiteren könnte eine Kombination von Word2Vec, Transformer-Ansatz und Doc2Vec für ontologiegestütztes Case-based Reasoning einen interessanten Forschungsansatz darstellen. Es könnte untersucht werden, ob durch einen kombinierten Einsatz zusätzliche Auswertemöglichkeiten bestehen, um beispielweise Vorhersagen von Projektentscheidungen zu treffen. In Bezug auf die Vorhersage von Projektentscheidungen unterscheidet sich der Ansatz von der Wiederverwendung von Erfahrungswissen dadurch, dass möglicherweise einzelne Projektentscheidungen auf Basis vergangener Projekterfahrungen vorhergesagt werden können. In diesem Fall liegt der Fokus auf einzelnen Projektentscheidungen und nicht auf dem gesamten Projekt. Die Vorhersagemöglichkeit mittels des Word2Vec-Algorithmus ist bereits Gegenstand von ersten Untersuchungen. So wird der Word2Vec-Algorithmus beispielsweise für die Entwicklung von Modellen zur Vorhersage des Schweregrads von Softwarefehlern genutzt; vgl. AGRAWAL/GOYAL (2021), S. 106-108. Hinsichtlich der Verbesserung von Einkaufsprognosen mittels des Word2Vec-Algorithmus siehe ESMELI/BADER-EL-DEN/ABDULLAHI (2020), S. 2-5. Zur Erkennung von Anomalien in Systemlogs mittels des Word2Vec-Algorithmus siehe WANG et al. (2022), S. 1210-1220.

Word2Vec könnte auch Möglichkeiten für die Definition von Adaptionsregeln, eine automatisierte Erzeugung von Ontologien und die Überprüfung einer Ontologie bieten. Erste Forschungsansätze existieren hierzu bereits. Für die mögliche automatisierte Erzeugung einer Ontologie mittels Word2Vec wird auf YOUN/NARAVANE/TAGKOPOULOS (2020), S. 2-3; MAHMOUD/ELBEH/ABDLKADER (2018), S. 184-188; WOHLGENANNT/MINIC (2016), S. 2-4, verwiesen. Die Arbeit von YOUN/NARA-VANE/TAGKOPOULOS wird beispielhaft vorgestellt, da die Autoren einen ähnlichen Ansatz wie in diesem Beitrag verwenden. Dabei wird ein Textkorpus (u. a. Wikipedia-Einträge und Lebensmitteldatenbanken) verwendet, um eine Ontologie für Lebensmittel zu erstellen; vgl. YOUN/NARAVANE/TAGKOPOULOS (2020), S. 2-3. Darüber hinaus wird auf CHEN et al. (2021), S. 1815-1843, verwiesen, die auf der Basis von Word2Vec den innovativen Ansatz OWL2Vec entwickeln, um eine automatisierte Ontologieeinbettung zu erzeugen.

Word2Vec könnte ein Werkzeug für die automatisierte Vorverarbeitung von Projektwissen aus natürlichsprachlichen Texten darstellen, um Attribut- und Relationswerte von Fällen zur Repräsentation sicherheitskritischer IT-Projekte automatisiert zu extrahieren. BEIßEL (2011), S. 219, hat diesen Aspekt bereits als möglichen Forschungsansatz angesprochen. Jedoch sah er Einschränkungen hinsichtlich der verfügbaren Textanalyse-Tools. Es wäre wünschenswert zu

überprüfen, ob sich die von BEIßEL genannten Einschränkungen mittels eines Word2Vec- oder Transformer-Ansatzes überwinden lassen.

Ein besonders gravierendes Problemfeld stellt die Adaption von Fällen im Rahmen des ontologiegestützten Case-based Reasonings dar. Zwar wurde hierfür in diesem Beitrag ein konzeptioneller Ansatz für den Erwerb von Adaptationswissen beschrieben. In einer weiterführenden Untersuchung wäre es jedoch wünschenswert, die Adaption von Fällen im Rahmen des ontologiegestützten Case-based Reasonings zu betrachten. Insbesondere die Implementierung von Adaptionsregeln, der automatisierte Erwerb von Adaptionsregeln und deren Anwendung stellen ein interessantes Forschungsfeld dar, in dem die hier verwendeten Techniken wie SWRL-Regeln, Word2Vec und der Transformer-Ansatz mögliche Unterstützung bieten könnten. Eine mögliche Unterstützung kann beispielsweise im Kontext von öffentlichen Vergabeverfahren (aus der Sicht eines potenziellen Auftragnehmers) darin bestehen, dass die Leistungsbeschreibungen auf einschlägigen Ausschreibungsplattformen automatisiert mittels des Transformer- oder Word2Vec-Ansatzes verarbeitet und mit vorhandenem Erfahrungswissen abgeglichen werden. Dies ermöglicht die Identifizierung von Ausschreibungen, bei denen eine höhere Gewinnwahrscheinlichkeit besteht, da bereits Erfahrungswissen vorliegt und Lösungsansätze für die ausgeschriebene Leistung vorhanden sind oder durch Anpassungsregeln an das neue Problem (die ausgeschriebene Leistung) angepasst werden können. Auf diese Weise kann der Vertriebsprozess eines Unternehmens gestärkt werden, indem eine Automatisierung sicherstellt, dass nur diejenigen Ausschreibungen bearbeitet werden, bei denen Erfahrungswissen vorliegt oder sogar Anpassungsregeln mittels SWRL, Word2Vec oder Transformer angewendet werden können, um das neue Problem zu lösen.

Literaturverzeichnis

AAMODT/PLAZA (1994)
Aamodt, A.; Plaza, E.: Case-Based Reasoning – Foundational Issues, Methodological Variations, and System Approaches. In: AI Communications, Vol. 7 (1994), No. 1, S. 39-59.

ABELS et al. (2006)
Abels, S.; Ahlemann, F.; Hahn, A.; Hausmann, K.; Strickmann, J. (2006): PROMONT – A Project Management Ontology as a Reference for Virtual Project Organizations. In: Hutchison, D.; Kanade, T.; Kittler, J.; Kleinberg, J. M.; Friedemann, M.; Mitchell, J. C.; Naor, M.; Nierstrasz, O.; Pandu Rangan, C.; Steffen, B.; Sudan, M.; Terzopoulos, D.; Tygar, D.; Vardi, M. Y.; Weikum, G.; Meersman, R.; Tari, Z.; Herrero, P. (Hrsg.): On the move to meaningful internet systems 2006: OTM 2006 workshops – OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET, OnToContent, ORM, PerSys, OTM Academy Doctoral Consortium, RDDS, SWWS, and SeBGIS 2006, Montpellier, France, October 29 - November 3, 2006, Proceedings, Part II. Berlin 2006, S. 813-823.

ABOU ASSALI et al. (2009)
Abou Assali, A.; Lenne, D.; Debray, B.: Case Retrieval in Ontology-Based CBR Systems. In: Mertsching, B.; Hund, M.; Aziz, Z. (Hrsg.): KI 2009 – Advances in Artificial Intelligence: 32nd Annual German Conference on AI, Paderborn, Germany, September 15-18, 2009, Proceedings. Berlin – Heidelberg (2009), S. 564-571.

AGRAWAL (2021)
Agrawal, T.: Hyperparameter Optimization in Machine Learning – Make Your Machine Learning and Deep Learning Models More Efficient. Berkeley 2021.

AGRAWAL/GOYAL (2021)
Agrawal, R.; Goyal, R.: Developing bug severity prediction models using word2vec. In: International Journal of Cognitive Computing in Engineering, Vol. 2 (2021), S. 104-115.

AICHELE (2021)
Aichele, C.: Künstliche Intelligenz für klein- und mittelständische Unternehmen. In: Aichele, C.; Herrmann, J. (Hrsg.): Betriebswirtschaftliche KI-Anwendungen – Digitale Geschäftsmodelle auf Basis Künstlicher Intelligenz. Wiesbaden 2021, S. 3-16.

AMAILEF/LU (2013)
Amailef, K.; Lu, J.: Ontology-supported case-based reasoning approach for intelligent m-Government emergency response services. In: Decision Support Systems, Vol. 55 (2013), Issue 1, S. 79-97.

AMAZON (2022)
Amazon: Annual Report 2021. Veröffentlicht am 03.02.2022. Online-Quelle verfügbar unter https://s2.q4cdn.com/299287126/files/doc_financials/2022/ar/Amazon-2021-Annual-Report.-pdf., zuletzt abgerufen am 11.04.2023.

AMAZON WEB SERVICES, INC. (2022a)
Amazon Web Services, Inc.: Was ist Python? – Python-Leitfaden für Cloud-Einsteiger – AWS. Copyright-Jahr 2022. Online-Quelle verfügbar unter https://aws.amazon.com/de/what-is/python/, zuletzt abgerufen am 11.12.2022.

AMAZON WEB SERVICES, INC. (2022b)
Amazon Web Services, Inc.: AWS Cloud9 Amazon Web Services. Copyright-Jahr 2022. Online-Quelle verfügbar unter https://aws.amazon.com/de/cloud9/, zuletzt abgerufen am 11.12. 2022.

AMAZON WEB SERVICES, INC. (2022c)
Amazon Web Services, Inc.: Customers. Copyright-Jahr 2022. Online-Quelle verfügbar unter https://aws.amazon.com/de/containers/customers/?customer-references-cards.sort-by=item.additionalFields.sortDate&customer-references-cards.sort-order=desc&awsf.customer-references-location=*all&awsf.customer-references-industry=*all&awsf.customer-references-segment=*all&awsf.customer-references-product=*all, zuletzt abgerufen am 11.12.2022.

AMAZON WEB SERVICES, INC. (2022d)
Amazon Web Services, Inc.: AWS Documentation. Copyright-Jahr 2022. Online-Quelle verfügbar unter https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do., zuletzt abgerufen am 11.12.2022.

AMAZON WEB SERVICES, INC. (2022e)
Amazon Web Services, Inc.: Globale AWS Infrastruktur & Availability Zones – AWS. Copyright-Jahr 2022. Online-Quelle verfügbar unter https://aws.amazon.com/de/about-aws/global-infrastructure/, zuletzt abgerufen am 11.12.2022.

ANGERMEIER (2016)
Angermeier, G.: PDCA-Zyklus. Online-Quelle unter https://www.projektmagazin.de/glossarterm/pdca-zyklus, zuletzt abgerufen am 11.04.2023.

ARAMO-IMMONEN (2009)
Aramo-Immonen, H.: Project Management Ontology – The Organizational Learning Perspective. Dissertation Tampere University of Technology 2009. Tampere 2009.

ASSALI/LENNE/DEBRAY (2010)
Assali, A. A.; Lenne, D.; Debray, B.: Heterogeneity in Ontological CBR Systems. In: Kacprzyk, J.; Montani, S.; Jain, L. C. (Hrsg.): Successful Case-based Reasoning Applications - I, Berlin – Heidelberg 2010, S. 97-116.

AVESANI/SUSI (2010)
Avesani, P.; Susi, A.: Case-Based Ranking for Environmental Risk Assessment. In: Kacprzyk, J.; Montani, S.; Jain, L. C. (Hrsg.): Successful Case-based Reasoning Applications – I. Berlin – Heidelberg 2010, S. 165-185.

AXELOS (2015)
Axelos: PRINCE2 Agile. Norwich 2015.

AXELOS (2018)
Axelos: Erfolgreiche Projekte managen mit PRINCE2. 6. Aufl. Norwich 2018.

AYYADEVARA (2018)
Ayyadevara, V. K.: Pro Machine Learning Algorithms – A Hands-On Approach to Implementing Algorithms in Python and R. New York 2018.

BAER/PURVES (2023)
Baer, M. F.; Purves, R. S.: Identifying Landscape Relevant Natural Language using Actively Crowdsourced Landscape Descriptions and Sentence-Transformers. In: KI – Künstliche Intelligenz, Vol. 37 (2023), Issue 1, S. 55-67.

BATSAKIS/TACHMAZIDIS/ANTONIOU (2017)
Batsakis, S.; Tachmazidis, I.; Antoniou, G.: Representing Time and Space for the Semantic Web. In: International Journal on Artificial Intelligence Tools, Vol. 26 (2017), No. 3, S. 1-34 (beitragsindividuelle Paginierung).

BEIßEL (2011)
Beißel, S.: Ontologiegestütztes Case-Based Reasoning – Entwicklung und Beurteilung semantischer Ähnlichkeitsindikatoren für die Wiederverwendung natürlichsprachlich repräsentierten Projektwissens. Dissertation Universität Duisburg-Essen 2011. Wiesbaden 2011.

BERGENRODT/KOWALSKI/ZELEWSKI (2015)
Bergenrodt, D.; Kowalski, M.; Zelewski, S.: Prototypische Implementierung des ontologiegestützten CBR-Tools jCORA. In: Zelewski, S.; Akca, N.; Kowalski, M. (Hrsg.): Organisatorische Innovationen mit Good Governance und Semantic Knowledge Management in Logistik-Netzwerken – Wissenschaftliche Grundlagen und Praxisanwendungen. Berlin 2015, S. 475-553.

BERGMANN/SCHAAF (2003)
Bergmann, R.; Schaaf, M.: Structural Case-Based Reasoning and Ontology-Based Knowledge Management – A Perfect Match? In: Journal of Universal Computer Science, Vol. 9 (2003), No. 7, S. 608-626.

BISONG (2019)
Bisong, E.: Building Machine Learning and Deep Learning Models on Google Cloud Platform – A Comprehensive Guide for Beginners. Berkeley 2019.

BÖGELSACK et al. (2022)
Bögelsack, A.; Chakraborty, U.; Kumar, D.; Rank, J.; Tischbierek, J.; Wolz, E.: SAP S/4 HANA-Systeme in Hyperscaler Clouds – Architektur, Betrieb und Setup von S/4HANA-Systemen in Microsoft Azure, Amazon Web Services und Google Cloud. Wiesbaden 2022.

BOUHANA et al. (2015)
Bouhana, A.; Zidi, A.; Fekih, A.; Chabchoub, H.; Abed, M.: An ontology-based CBR approach for personalized itinerary search systems for sustainable urban freight transport. In: Expert Systems with Applications, Vol. 42 (2015), Issue 1, S. 3724-3741.

BUNDESMINISTERIUM DES INNERN (2009)
Bundesministerium des Innern: Nationale Strategie zum Schutz Kritischer Infrastrukturen (KRITIS-Strategie). Berlin 2009.

CALDATO (2020)
Caldato, C.: Cloud Native for the Enterprise. Peking et al. 2020.

CAPGEMINI (2021)
Capgemini: IT-Trends Studie 2021. Veröffentlicht im Jahr 2021. Online-Quelle verfügbar unter https://www.capgemini.com/de-de/wp-content/uploads/sites/5/2021/02/Studie_IT-Trends_20-21_Capgemini.pdf, zuletzt abgerufen am 11.04.2023.

CAROLLA (2015)
Carolla, M.: Ein Referenz-Datenmodell für Campus-Management-Systeme in deutschsprachigen Hochschulen. Dissertation Universität Bielefeld 2014. Wiesbaden 2015.

CASS (2022)
Cass, S.: Top Programming Languages 2022. Veröffentlicht im Jahr 2022. Online-Quelle verfügbar unter https://spectrum.ieee.org/top-programming-languages-2022, zuletzt abgerufen am 11.04.2023.

CHEN et al. (2021)
Chen, J.; Hu, P.; Jimenez-Ruiz, E.; Holter, O. M.; Antonyrajah, D.; Horrocks, I.: OWL2Vec*: embedding of OWL ontologies. In: Machine Learning, Vol. 110 (2021), S. 1813-1845.

DEBELLIS (2021)
DeBellis, M.: A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins. O. O. 2021.

DELOITTE (2022)
Deloitte: TechTrends 2022. Veröffentlicht im Jahr 2022. Online-Quelle verfügbar unter https://www2.deloitte.com/content/dam/Deloitte/pt/Documents/tech-trends/tech-trends-2022/-DI_Tech-trends-2022.pdf, zuletzt abgerufen am 11.04.2023.

DER BUNDESRAT DER SCHWEIZER REGIERUNG (2021)
Der Bundesrat der Schweizer Regierung: Herausforderungen bei zwei wichtigen Projekten zur sicheren Kommunikation. Veröffentlicht am 16.02.2021 Online-Quelle verfügbar unter https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-82340.html, zuletzt abgerufen am 11.04.2023.

DEVLIN et al. (2019)
Devlin, J.; Chang, M.-W.; Lee, K./Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Burstein, J.; Doran, C.; Solorio, T. (Hrsg.): Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis 2019, S. 4171-4186.

DI GENNARO/BUONANNO/PALMIERI (2021)
Di Gennaro, G.; Buonanno, A.; Palmieri, F. A. N.: Considerations about learning Word2Vec. In: The Journal of Supercomputing, Vol. 77 (2021), Issue 11, S. 12320-12335.

DI MARTINO et al. (2022)
Di Martino, B.; Bombace, V.; Colucci Cante, L.; Esposito, A.; Graziano, M.; Pezzullo, G. J.; Tofani, A.; D'Agostino, G.: Machine Learning, Big Data Analytics and Natural Language Processing Techniques with Application to Social Media Analysis for Energy Communities. In: Barolli, L. (Hrsg.): Complex, Intelligent and Software Intensive Systems – Proceedings of the 16th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2022). Cham 2022, S. 425-434.

DIVARI (2011)
Divari, V.: Konzeption und Entwicklung eines Verfahrens zum Matching von Prozessmodellen. Veröffentlicht am 18.07.2011. Online-Publikation verfügbar unter https://elib.uni-stuttgart.de/bitstream/11682/2754/1/DIP_3149.pdf, zuletzt abgerufen am 11.04.2023.

DONG/HUSSAIN/CHANG (2011)
Dong, H.; Hussain, F.; Chang, E.: ORPMS – An Ontology-based Real-time Project Monitoring System in the Cloud. In: Journal of Universal Computer Science, Vol. 17 (2011), No. 8, S. 1161-1182.

DOWNEY (2021)
Downey, A. B.: Think Python – Systematisch programmieren lernen mit Python. Heidelberg 2021.

DUARTE/BELO (2023)
Duarte, A.; Belo, O.: Blending Case-Based Reasoning with Ontologies for Adapting Diet Menus and Physical Activities. In: Arai, K. (Hrsg.): Intelligent Systems and Applications: Proceedings of the 2022 Intelligent Systems Conference (IntelliSys) Volume 1. Cham 2023, S. 829-843.

EL-AMIR/HAMDY (2020)
El-Amir, H.; Hamdy, M.: Deep Learning Pipeline – Building a Deep Learning Model with TensorFlow. Berkeley 2020.

EL JERROUDI (2010)
El Jerroudi, Z.: Eine interaktive Vorgehensweise für den Vergleich und die Integration von Ontologien. Lohmar 2010.

EMMENEGGER et al. (2017)
Emmenegger, S.; Hinkelmann, K.; Laurenzi, E.; Martin, A.; Thönssen, B.; Witschel, H. F.; Zhang, C.: An Ontology-Based and Case-Based Reasoning Supported Workplace Learning Approach. In: Hammoudi, S.; Ferreira Pires, L.; Selic, B.; Desfray, P. (Hrsg.): Model-Driven Engineering and Software Development – 4th International Conference, MODELSWARD 2016, Rome, Italy, February 19-21, 2016, Revised Selected Papers. Cham 2017, S. 333-354.

ERNE (2019)
Erne, R.: Lean Project Management – Wie man den Lean-Gedanken im Projektmanagement einsetzen kann. Wiesbaden 2019.

ESMELI/BADER-EL-DEN/ABDULLAHI (2020)
Esmeli, R.; Bader-El-Den, M.; Abdullahi, H.: Using Word2Vec Recommendation for Improved Purchase Prediction. In: IEEE: 2020 International Joint Conference on Neural Networks (IJCNN). Piscataway 2020, S. 1-8 (eigene Paginierung).

EUROSTAT (2022)
Eurostat: Statistical regions in the European Union and partner countries – NUTS and statistical regions 2021. 2022 re-edition. Luxemburg 2022.

FORTUNE BUSINESS INSIGHTS (2023)
Fortune Business Insights: Cloud Computing Market Size, Share & Covid-19 Impact Analysis, By Type (Public Cloud, Private Cloud, Hybrid Cloud), By Service (Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS)), By Industry (BFSI, IT and Telecommunications, Government, Consumer Goods and Retail, Healthcare, Manufacturing and Others), and Regional Forecast, 2023-203. Veröffentlicht im Mai 2020. Online-Quelle verfügbar unter https://www.fortunebusinessinsights.com/cloud-computing-market-102697, zuletzt abgerufen am 11.06.2023.

FOWLER/LEWIS (2015)
Fowler, M.; Lewis, J.: Microservices – Nur ein weiteres Konzept in der Softwarearchitektur oder mehr? In: Objektspektrum, Heft 1/2015, S. 14-21.

FRANK/SCHUMACHER/TAMM (2019)
Frank, R.; Schumacher, G.; Tamm, A: Cloud-Transformation – Wie die Public Cloud Unternehmen verändert. Wiesbaden 2019.

FRITZSCH et al. (2019)
Fritzsch, J.; Bogner, J.; Zimmermann, A.; Wagner, S.: From Monolith to Microservices – A Classification of Refactoring Approaches. In: Bruel, J. M.; Mazzara, M.; Meyer, B. (Hrsg.): Software Engineering Aspects of Continuous Development and New Paradigms: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers. Cham 2019, S. 128-141.

FROCHTE (2021)
Frochte, J.: Maschinelles Lernen: Grundlagen und Algorithmen in Python. München 2021.

GARTNER (2021)
Gartner: Top Strategic Technology Trends for 2022. Copyright 2021. Online-Quelle verfügbar unter https://www.technova-cpi.org/images/Documenti-pdf/Top%20Strategic%20Technology%20Trends%20for%202022_Gartner_31gen2022.pdf, zuletzt abgerufen am 11.04.2023.

GARTNER (2022a)
Gartner: Magic Quadrant for Cloud AI Developer Services. Veröffentlicht am 23.05.2022. Online-Quelle verfügbar unter https://www.gartner.com/en/documents/4014812, zuletzt abgerufen am 11.04.2023.

GARTNER (2022b)
Gartner: Magic Quadrant for Cloud Infrastructure and Platform Services. Copyright 2022. Online-Quelle verfügbar unter https://www.gartner.com/technology/media-products/reprints/-AWS/1-271W1OSP-DEU.html, zuletzt abgerufen am 09.12.2022.

GASSMANN (2001)
Gassmann, O.: High-Risk-Projekte als Erfolgsfaktor in dynamischen Industrien. In: Gassmann, O.; Kobe, C.; Voit, E. (Hrsg.): High-Risk-Projekte. Berlin – Heidelberg 2001, S. 3-23.

GITHUB (2022a)
GitHub: GitHub - RaRe-Technologies/genism – Topic Modelling for Humans. Veröffentlicht im Jahr 2022. Online-Quelle verfügbar unter https://github.com/RaRe-Technologies/gensim, zuletzt abgerufen am 17.12.2022.

GITHUB (2022b)
GitHub: GitHub – RaRe-Technologies/gensim-data: Data repository for pretrained NLP models and NLP corpora. Veröffentlicht im Jahr 2022. Online-Quelle verfügbar unter https://github.com/RaRe-Technologies/gensim-data, zuletzt abgerufen am 17.12.2022.

GLEB (2021)
Gleb, T.: Systematic Cloud Migration – A Hands-On Guide to Architecture, Design, and Technical Implementation. Thornhill 2021.

GOLDBERG/LEVY (2014)
Goldberg, Y.: Levy, O.: word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. Online-Quelle verfügbar unter https://arxiv.org/abs/1402.3722, zuletzt abgerufen am 11.04.2023.

GÓMEZ-PÉREZ/FERNÁNDEZ-LÓPEZ/CORCHO (2004)
Gómez-Pérez, A.; Fernández-López, M.; Corcho, O.: Ontological Engineering – With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. London 2004.

GONIWADA (2022)
Goniwada, S. R.: Cloud Native Architecture and Design – A Handbook for Modern Day Architecture and Design with Enterprise-Grade Examples. New York 2022.

GOOGLE (2022)
Google: Google Colaboratory. Copyright 2022. Online-Quelle verfügbar unter https://colab.research.google.com/, zuletzt abgerufen am 11.04.2023.

GOOGLE CLOUD (2020)
Google Cloud: Das neue Grace-Hopper-Seekabel zwischen den USA, dem Vereinigten Königreich und Spanien. Veröffentlicht am 21.07.2020. Online-Quelle verfügbar unter https://cloud.google.com/blog/de/products/infrastruktur/das-neue-grace-hopper-seekabel-von-google, zuletzt abgerufen am 29.07.2023.

GOOGLE CLOUD (2022a)
Google Cloud: Produkte und Dienste der Google Cloud. Copyright 2022. Online-Quelle verfügbar unter https://cloud.google.com/products, zuletzt abgerufen am 19.12.2022.

GOOGLE CLOUD (2022b)
Google Cloud: Weltweite Standorte – Regionen und Zonen der Google Cloud. Copyright 2022. Online-Quelle verfügbar unter https://cloud.google.com/about/locations?hl=de#regions, zuletzt abgerufen am 12.19.2022.

GOOGLE DEVELOPERS (2022)
Google Developers: Python. Copyright 2022. Online-Quelle verfügbar unter https://developers.google.com/learn/topics/python, zuletzt abgerufen am 11.12.2022.

GRUBER (1993)
Gruber, T. R.: A translation approach to portable ontology specifications. In: Knowledge Acquisition, Vol. 5 (1993), S. 199-220.

GUO/HU/PENG (2012)
Guo, Y.; Hu, J.; Peng, Y.: A CBR system for injection mould design based on ontology – A case study. In: Computer-Aided Design, Vol. 44 (2012), Issue 6, S. 496-508.

GUSKOV et al. (2022)
Guskov, G.; Zarayskiy, V.; Filippov, A.; Romanov, A.: An Approach to Ontology-Based Smart Search in E-commerce. In: Golenkov, V.; Krasnoproshin, V.; Golovko, V.; Shunkevich, D. (Hrsg.): Open Semantic Technologies for Intelligent Systems – 11th International Conference, OSTIS 2021, Minsk, Belarus, September 16–18, 2021, Revised Selected Papers. Cham 2022, S. 361-372.

HABERMANN (2013)
Habermann, F.: Hybrides Projektmanagement – agile und klassische Vorgehensmodelle im Zusammenspiel. In: HMD Praxis der Wirtschaftsinformatik, Vol. 50 (2013), S. 93-102.

HARRIS (2022)
Harris, C.: Microservices und monolithische Architektur im Vergleich. Veröffentlicht im Jahr 2022. Online-Quelle verfügbar unter https://www.atlassian.com/de/microservices/microservices-architecture/microservices-vs-monolith, zuletzt abgerufen am 11.04.2023.

HENNEBERGER (2016)
Henneberger, M.: Von „Cloud Enabling“ zu „Cloud Native“ – Wie Cloud Computing die Unternehmens-IT verändert. In: Wirtschaftsinformatik & Management, Vol. 8 (2016), Issue 5, S. 8-19.

HERDER/ZELEWSKI/SCHAGEN (2022)
Herder, M.; Zelewski, S.; Schagen, J. P.: Evaluation des Prototyps jCORA im Rahmen des KI-LiveS-Projekts hinsichtlich Anforderungen an die „intelligente“ Wiederverwendung von Erfahrungswissen im Projektmanagementbereich. Arbeitsbericht Nr. 57, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), zugleich KI-LiveS-Projektbericht Nr. 11. Essen 2022.

HILMER/KRIEG (2014)
Hilmer, S.: Krieg, A.: Standardisierung vs. Kultur: Klassisches und agiles Projektmanagement im Vergleich. In: Engstler, M.; Hanser, E.; Mikusz, M.; Herzwurm, G. (Hrsg.): Projektmanagement und Vorgehensmodelle 2014 – Soziale Aspekte und Standardisierung. Bonn 2014, S. 47-57.

HOWARD/RUDER (2018)
Howard, J.; Ruder, S.: Universal Language Model Fine-tuning for Text Classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), Melbourne, Australia, July 15- 20. Stroudsburg 2018, S. 328-339.

HUGHES (2010)
Hughes, R.: Project Management Process Ontologies – A Proof of Concept. In: UK Academy for Information Systems Conference, Spring 23.03.2010, Proceedings. o. O. 2010, Beitrag 30, S. 1-19 (beitragsindividuelle Paginierung).

INITIATIVE D21 (2022)
Initiative D21: D21-Digital-Index 2021/2022: Jährliches Lagebild zur Digitalen Gesellschaft. Vertiefungsthema – Digitale Nachhaltigkeit. Online-Quelle verfügbar unter https://initiatived21.de/app/uploads/2022/02/d21-digital-index-2021_2022.pdf, zuletzt abgerufen am 11.04.2023.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2000)
International Organization for Standardization: ISO/IEC 9126-1. Copyright 2000 – Information technology – Software product quality. Part1: Quality model. Online-Quelle verfügbar unter https://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-1%20Standard.pdf, zuletzt abgerufen am 11.04.2023.

JÄGER/RIEKEN/ERNST (2020)
Jäger, H. A.; Rieken, R. O. G.; Ernst, E.: Herausforderung Datenschutz und Datensicherheit in der Cloud. In: Jäger, H. A.; Rieken, R. O. G. (Hrsg.): Manipulationssichere Cloud-Infrastrukturen – Nachhaltige Digitalisierung durch Sealed Cloud Security. Wiesbaden 2020, S. 3-31.

JAMSHIDI et al. (2018)
Jamshidi, P.; Pahl, C.; Mendonca, N. C.; Lewis, J.; Tilkov, S.: Microservices – The Journey So Far and Challenges Ahead. In: IEEE Software, Vol. 35 (2018), Issue 3, S. 24-35.

JI/PARK/LEE (2012)
Ji, S.-H.; Park, M.; Lee, H.-S.: Case Adaptation Method of Case-Based Reasoning for Construction Cost Estimation in Korea. In: Journal of Construction Engineering and Management, Vol. 138 (2012), No. 1, S. 43-52.

KALS (2021)
Kals, J.: Qualitätsmanagement (QM). In: Wannenwetsch, H. (Hrsg.): Integrierte Materialwirtschaft, Logistik, Beschaffung und Produktion – Supply Chain im Zeitalter der Digitalisierung. Berlin 2021, S. 273-321.

KANTARWORLDPANEL (2022)
Kantarworldpanel: Android vs. iOS – Smartphone OS sales market share evolution. Copyright 2022. Online-Quelle verfügbar unter https://www.kantarworldpanel.com/global/smartphone-os-market-share/, zuletzt abgerufen am 19.12.2022.

KIM et al. (2012)
Kim, M.; Lee, S.; Woo, S.; Shin, D. H.: Approximate cost estimating model for river facility construction based on case-based reasoning with genetic algorithms. In: KSCE Journal of Civil Engineering, Vol. 16 (2012), No. 3, S. 283-292.

KIM/SHIM (2014)
Kim, S.; Shim, J. H.: Combining case-based reasoning with genetic algorithm optimization for preliminary cost estimation in construction industry. In: Canadian Journal of Civil Engineering, Vol. 41 (2014), No. 1, S. 65-73.

KLEIN (2021)
Klein, B.: Einführung in Python 3 – Für Ein- und Umsteiger. München 2021.

KLÜVER/KLÜVER (2021)
Klüver, C.; Klüver, J.: Teil I: KI – Das Self-Enforcing Network (SEN). In: Klüver, C.; Klüver, J. (Hrsg.): Neue Algorithmen für praktische Probleme – Variationen zu Künstlicher Intelligenz und Künstlichem Leben. Wiesbaden 2021, S. 9-20.

KLÜVER/KLÜVER/SCHMIDT (2021)
Klüver, C.; Klüver, J.: Schmidt, J.: Modellierung komplexer Prozesse durch naturanaloge Verfahren – Künstliche Intelligenz und Künstliches Leben. 3. Aufl. Wiesbaden 2021.

KNEUPER (2021)
Kneuper, R.: Datenschutz für Softwareentwicklung und IT – Eine praxisorientierte Einführung. Berlin – Heidelberg 2021.

KRATZKE (2022)
Kratzke, N.: Cloud-native Computing – Software Engineering von Diensten und Applikationen für die Cloud. München 2022.

KUHN (2023)
Kuhn, T: Datendiebstähle bei Cloud-Anbieter – Sicherheit in der Cloud ist nur eine Illusion! Veröffentlicht am 24.08.2023. Online-Quelle verfügbar unter https://www.wiwo.de/technologie/digitale-welt/datendiebstaehle-bei-cloudanbietern-sicherheit-in-der-cloud-ist-nur-eine-illusion/29352358.html, zuletzt abgerufen am 11.09.2023.

KULKARNI/SHIVANANDA (2021)
Kulkarni, A.; Shivananda, A.: Natural Language Processing Recipes – Unlocking Text Data with Machine Learning and Deep Learning using Python. New York 2021.

KUNSCHKE/SPITZ/POHLE (2022)
Kunschke, D.; Spitz, M. F.; Pohle, J.: KI in der Cloud – Proprietäre Services vs. Open-Source-Technologien. In: Kunschke, D.; Spitz, M. F.; Pohle, J. (Hrsg.): FinTech – Digitalisierung, Künstliche Intelligenz und aufsichtsrechtliche Regulierung von Finanzdienstleistungen. 2. Aufl., Berlin, S. 391-408.

LAMY (2017)
Lamy, J.-B.: Owlready – Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. In: Artificial Intelligence in Medicine, Vol. 80 (2017), S. 11-28.

LAMY (2021)
Lamy, J.-B.: Ontologies with Python – Programming OWL 2.0 Ontologies with Python and Owlready2. Berkeley 2021.

LAMY (2023)
Lamy, J.-B.: Owlready2 Documentation. Release 0.41. Veröffentlicht am 04.04.2023. Online-Quelle verfügbar https://readthedocs.org/projects/owlready2/downloads/pdf/latest/, zuletzt abgerufen am 08.04.2023.

LIEDTKE (2022)
Liedtke, T.: Informationssicherheit – Möglichkeiten und Grenzen. Berlin – Heidelberg 2022.

LIN et al. (2012)
Lin, Y.; Hilaire, V.; Gaud, N.; Koukam, A.: Scrum Conceptualization Using K-CRIO Ontology. In: Aberer, K.; Damiani, E.; Dillon, T. (Hrsg.): Data-Driven Process Discovery and Analysis. Berlin – Heidelberg 2012, S. 189-211.

LINTHICUM (2022)
Linthicum, D.: 3 Gründe – Warum Sie mit der Cloud kein Geld sparen. Veröffentlicht am 30.09.2022. Online-Quelle verfügbar unter https://www.cio.de/a/warum-sie-mit-der-cloud-kein-geld-sparen,3612872, zuletzt abgerufen am 11.04.2023.

LÜNENDONK (2021)
Lünendonk: Cloud-native Software Development – Mit Cloud-Technologien und Agilität zu mehr Innovationsgeschwindigkeit und Wettbewerbsvorteilen. Lünendonk-Studie. Veröffentlicht im Jahr 2021. Online-Quelle verfügbar unter https://www.luenendonk.de/produkte/studien-publikationen/luenendonk-studie-2021-cloud-native-software-development-it/, zuletzt abgerufen am 11.04.2023.

LUONG/PHAM/MANNING (2015)
Luong, M.-T.; Pham, H.; Manning, C. D.: Effective Approaches to Attention-based Neural Machine Translation. Veröffentlicht am 20.09.2015. Online-Quelle verfügbar unter https://arxiv.org/abs/1508.04025, zuletzt abgerufen am 05.05.2023.

LUTZ (2007)
Lutz, M.: Einführung in Python. Köln – Boston – Massachusetts 2007.

MAHMOUD/ELBEH/ABDLKADER (2018)
Mahmoud, N.; Elbeh, H.; Abdlkader, H. M.: Ontology Learning Based on Word Embeddings for Text Big Data Extraction. In: IEEE: ICENCO 2018: 14th International Computer Engineering Conference „Secure Smart Societies“. Computer Engineering Department Faculty of Engineering, Cairo University Giza, Ägypten, 29.-30. Dezember 2018. Piscataway 2018, S. 183-188.

MARTIN et al. (2017)
Martin, A.; Emmenegger, S.; Hinkelmann, K.; Thönssen, B.: A viewpoint-based case-based reasoning approach utilising an enterprise architecture ontology for experience management. In: Enterprise Information Systems, Vol. 11 (2017), Issue 4, S. 551-575.

MATZKA (2021)
Matzka, S.: Künstliche Intelligenz in den Ingenieurwissenschaften – Maschinelles Lernen verstehen und bewerten. Wiesbaden 2021.

MAWLOOD-YUNIS (2022)
Mawlood-Yunis, A.-R.: Android for Java Programmers. Cham 2022.

MEND (2018)
Mend: What are the most secure programming languages? Copyright 2023. Online-Quelle verfügbar unter https://www.mend.io/most-secure-programming-languages/, zuletzt abgerufen am 11.04.2023.

MICROSOFT (2023a)
Microsoft: Was sind Azure-Regionen und -Verfügbarkeitszonen? Veröffentlicht am 16.03.2023. Online-Quelle verfügbar unter https://learn.microsoft.com/de-de/azure/availability-zones/az-overview, zuletzt abgerufen am 11.04.2023.

MICROSOFT (2023b)
Microsoft: Azure-Produkte. Azure Produkte suchen oder durchsuchen. Copyright 2023. Online-Quelle verfügbar unter https://azure.microsoft.com/de-de/products/, zuletzt abgerufen am 11.04.2023.

MIKOLOV et al. (2013a)
Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.: Efficient Estimation of Word Representations in Vector Space. Veröffentlicht am 07.09.2013. Online-Quelle verfügbar unter https://arxiv.org/abs/1301.3781, zuletzt abgerufen am 05.05.2023.

MIKOLOV et al. (2013b)
Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. Veröffentlicht am 16.10.2013. Online-Quelle verfügbar unter https://arxiv.org/abs/1310.4546, zuletzt abgerufen am 05.05.2023.

MIKOLOV/YIH/ZWEIG (2013)
Mikolov, T.; Yih, W.; Zweig, G.: Linguistic regularities in continuous space word representations. In: Vanderwende, L., Hal Daumé III, H.; , Kirchhoff, K. (Hrsg.): The 2013 Conference of the North American Chapter of the Association for Computational Linguistics – HumanLanguageTechnologies, 9-14 June 2013, Westin Peachtree Plaza Hotel Atlanta, Georgia. Stroudsburg 2013, S. 746-751.

MIRFENDRESKI (2022)
Mirfendreski, A.: Künstliche Intelligenz für die Entwicklung von Antrieben – Historie, Arbeitsprozesse, Konzepte, Methoden und Anwendungsbeispiele. Berlin 2022.

MOHD/DHASMANA/UPADHYAY (2021)
Mohd, N.; Dhasmana, G.; Upadhyay, D.: Implementation of Traditional Vs. Transformer Machine Learning Models. In: Webology, Vol. 18 (2021), No. 4, S. 2392-2399.

NIGHOJKAR/KHLYZOVA/LICATO (2022)
Nighojkar, A.; Khlyzova, A.; Licato, J.: Cognitive Modeling of Semantic Fluency Using Transformers. Veröffentlicht am 20.08.2022. Online-Quelle verfügbar unter https://arxiv.org/abs/2208.09719, zuletzt abgerufen am 05.05.2023.

NKISI-ORJI et al. (2020)
Nkisi-Orji, I.; Wiratunga, N.; Palihawadana, C.; Recio-García, J. A.; Corsar, D.: Cloud CBR: Towards Microservices Oriented Case-Based Reasoning: In: Watson, I.; Weber, R. (Hrsg.): Case-Based Reasoning Research and Development. Cham 2020, S. 129-143.

NKISI-ORJI et al. (2022)
Nkisi-Orji, I.; Palihawadana, C.; Wiratunga, N.; Corsar, D.; Wijekoon, A.: Adapting Semantic Similarity Methods for Case-Based Reasoning in the Cloud. In: Keane, M. T.; Wiratunga, N. (Hrsg.): Case-Based Reasoning Research and Development. Cham 2022, S. 125-139.

NOY/MCGUINNESS (2001)
Noy, F.; McGuinness, D.: Ontology Development 101 – A Guide to Creating Your First Ontology. Veröffentlicht im Jahr 2001. Online-Quelle verfügbar unter https://protege.stanford.edu/publications/ontology_development/ontology101.pdf, zuletzt abgerufen am 11.04.2023.

NUGROHO et al. (2023)
Nugroho, Y. S.; Islam, S.; Gunawan, D.; Kurniawan, Y. I.; Hossain, M. J.; Kabir, M. H.: A Study of E-commerce Platform Issues Shared by Developers on Stack Overflow. In: Dutta, P.; Bhattacharya, A.; Dutta, S.; Lai, W.-C. (Hrsg.): Emerging Technologies in Data Mining and Information Security – Proceedings of IEMIS 2022, Volume 1. Singapore 2023, S. 291-299.

OBEID et al. (2022)
Obeid, C.; Lahoud, C.; El Khoury, H.; Champin, P.-A.: A novel Hybrid Recommender System Approach for Student Academic Advising Named COHRS, Supported by Case-based Reasoning and Ontology. In: Computer Science and Information Systems, Vol. 19 (2022), No. 2, S. 979-1005.

OLIVEIRA ROCHA (2021)
Oliveira Rocha, H. F.: Practical Event-Driven Microservices Architecture – Building Sustainable and Highly Scalable Event-Driven Microservices. Berkeley 2021.

PAAß/HECKER (2020)
Paaß, G.; Hecker, D.: Künstliche Intelligenz – Was steckt hinter der Technologie der Zukunft? Wiesbaden 2020.

PANDAY/SAHU (2023)
Panday, M.; Sahu, S.: Topic Modelling Based Semantic Search. In: Sharma, N.; Chakrabarti, A.; Balas, V. E. (Hrsg.): Data Management, Analytics and Innovation – Proceedings of ICDMAI 2022. Singapore 2023, S. 291-302.

PERTLWIESER (2022)
Pertlwieser, M.: Das richtige Digitalisieren – Eine 'Masterclass' zum digitalen Wandel für Manager:innen und Unternehmer:innen. Wiesbaden 2022, S. 25-50.

PFITZINGER/JESTÄDT (2017)
Pfitzinger, B.; Jestädt, T.: IT-Betrieb: Management und Betrieb der IT in Unternehmen – Grundlagen, Statistik und maschinelles Lernen. Berlin – Heidelberg 2017.

POPULARITY OF PROGRAMMING LANGUAGE (2023)
PopularitY of Programming Language – PYPL PopularitY of Programming Language index. Copyright 2023. Online-Quelle verfügbar unter https://pypl.github.io/PYPL.html, zuletzt abgerufen am 11.04.2023.

POTHECARY (2021)
Pothecary, R.: Running Microsoft Workloads on AWS – From active directory, databases, development, and beyond. New York 2021.

PYTHON (2002)
Python: Mission Statement. Veröffentlicht im Jahr 2002. Online-Quelle verfügbar unter https://www.python.org/psf/mission/, zuletzt abgerufen am 11.04.2023.

PYTHON (2023a)
Python: Python Software Foundation. Copyright 2023. Online-Quelle verfügbar unter https://www.python.org/psf/, zuletzt abgerufen am 11.04.2023.

PYTHON (2023b)
Python: What is Python? Executive Summary. Copyright 2023. Online-Quelle verfügbar unter https://www.pyhon.org/doc/essays/blurb/, zuletzt abgerufen am 11.04.2023.

RADZIEJOWSKA/ZIMA (2015)
Radziejowska, A.; Zima, K.: The Concept of a Knowledge Base to Aid in Cost Estimating of Sports Facilities. In: International Journal of Contemporary Management, Vol. 14 (2015), No. 3, S. 99-113.

RASCHKA/MIRJALILI (2019)
Raschka, S.; Mirjalili, V.: Python Machine Learning – Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2. 3. Auflage. Birmingham 2019.

RASHID (2017)
Rashid, T.: Neuronale Netze selbst programmieren – Ein verständlicher Einstieg mit Python. Heidelberg 2017.

RAU (2016)
Rau, K.-H.: Agile objektorientierte Software-Entwicklung – Schritt für Schritt vom Geschäftsprozess zum Java-Programm. Wiesbaden 2016.

RECHLIN (2004)
Rechlin, S.: Die deutschen Kommunen im Mehrebenensystem der Europäischen Union – Betroffene Objekte oder aktive Subjekte? Discussion Paper SP IV 2004-101. Wissenschaftszentrum Berlin für Sozialforschung. Berlin 2004.

RECIO-GARCÍA et al. (2007)
Recio-García, J. A.; Díaz-Agudo, B.; González-Calero, P.; Sánchez-Ruiz-Granados, A.: Ontology based CBR with jCOLIBRI. In: Ellis, R.; Allen, T.; Tuson, A. (Hrsg.): Applications and Innovations in Intelligent Systems XIV. London 2007, S. 149-162.

RECIO-GARCÍA/GONZÁLEZ-CALERO/DÍAZ-AGUDO (2014)
Recio-García, J. A.; González-Calero, P. A.; Díaz-Agudo, B.: JCOLIBRI2: A framework for building Case-based reasoning systems. In: Science of Computer Programming, Vol. 79 (2014), S. 126-145.

REGENFUß/NINK (2022)
Regenfuß, T./Nink, T.: Was kann die Google Cloud? Veröffentlicht am 05.10.2022. Online-Quelle verfügbar unter https://www.cio.de/a/was-kann-die-google-cloud,3667814#:~:text=In%20der%20Google%20Cloud%20erfolgt,kostenoptimierten%20Zuschnitt%20der%20virtuellen%20Server, zuletzt abgerufen am 11.04.2023.

ŘEHŮŘEK (2011)
Řehůřek, R.: Scalability of semantic analysis in natural language processing. Dissertation Masaryk University. Brünn 2011.

ŘEHŮŘEK (2022)
Řehůřek, R.: API Reference. Word2vec embeddings. Veröffentlicht am 21.12.2022. Online-Quelle verfügbar unter https://radimrehurek.com/gensim/models/word2vec.html, zuletzt abgerufen am 11.04.2023.

ŘEHŮŘEK/SOJKA (2010)
Řehůřek, R.; Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of LREC 2010 Workshop New Challenges for NLP Frameworks in Valletta, Malta. Valletta 2010, S. 46-50.

RICHTER (2008)
Richter, M.: Similarity. In: Perner, P. (Hrsg.): Case-Based Reasoning on Images and Signals. Berlin – Heidelberg (2008), S. 25-90.

RONG (2014)
Rong, X.: word2vec Parameter Learning Explained. Veröffentlicht am 11.11.2014. Online-Quelle verfügbar unter https://arxiv.org/abs/1411.2738, zuletzt abgerufen am 05.05.2023.

SANTOS JÚNIOR et al (2021)
Santos Júnior, P. S.; Barcellos, M. P.; Falbo, R. d. A.; Almeida, J. P. A.: From a Scrum Reference Ontology to the Integration of Applications for Data-Driven Software Development. In: Information and Software Technology, Vol. 136 (2021), Article 106570, S. 1-27.

SARANTIS/ASKOUNIS (2009)
Sarantis, D.; Askounis, D.: A project management ontology as a reference for e-Government projects. In: International Conference for Internet Technology and Secured Transactions, 2009. ICITST 2009; 9-12 Nov. 2009. Piscataway 2009, o. S. (beitragsindividuelle Paginierung S. 1-8).

SARKAR (2019)
Sarkar, D.: Text analytics with Python – A practitioner's guide to natural language processing. New York 2019.

SCHAGEN et al. (2022)
Schagen, T.; Heeb, T.; Zelewski, S.; Schagen, J. P.: Entwicklung eines E-Learning-Moduls für ein ontologiegestütztes Case-based Reasoning Tool für das betriebliche Projektmanagement. Arbeitsbericht Nr. 54, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), zugleich KI-LiveS-Projektbericht Nr. 8. Essen 2022.

SCHWABER/SUTHERLAND (2020)
Schwaber, K.; Sutherland, J.: The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game. Veröffentlicht im November 2020. Online-Quelle verfügbar unter https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf, zuletzt abgerufen am 06.07.2024.

SENATSVERWALTUNG FÜR INNERES UND SPORT (2016)
Senatsverwaltung für Inneres und Sport: Antrag auf Freigabe von nach § 24 Abs. 3 LHO gesperrt veranschlagten Ausgaben bei Baumaßnahmen Kapitel 1250, Titel 70160 – Polizei und Feuerwehr; Neubau einer Kooperativen Leitstelle auf dem Gelände Gallwitzallee, Feuerwehrleitstelle Nikolaus-Groß-Weg, Errichtung eines Erweiterungsbaus und Sanierung des Bestandsgebäudes hier: Ausschreibung und Beauftragung des Systemlieferanten sowie der projektbegleitenden Qualitätssicherung. Veröffentlicht am 19.08.2016. Online-Quelle verfügbar unter https://www.parlament-berlin.de/ados/17/Haupt/vorgangh17-2941-v.pdf, zuletzt abgerufen am 11.04.2023

Sethupathy (2024)
Sethupathy, G.: Künstliche Intelligenz für das Wissensmanagement von sicherheitskritischen IT-Projekten – Ontologiegestütztes Case-based Reasoning zur „intelligenten“ Wiederverwendung von Erfahrungswissen. Dissertation 2024, Fakultät für Wirtschaftswissenschaft der Universität Duisburg-Essen. Berlin 2024.

SHEEBA/KRISHNAN/BERNARD (2012)
Sheeba, T.; Krishnan, R.; Bernard, M.: An Ontology in Project Management Knowledge Domain. In: International Journal of Computer Applications, Vol. 56 (2012), No. 5, o. S. (beitragsindividuelle Paginierung S. 1-7).

SLASHDATA (2021)
SlashData: State-of-Cloud-Native-Development 2021 – The latest trends from our Q3 2021 survey of 19000 developers. Veröffentlicht im Jahr 2021. Online-Quelle verfügbar unter https://www.cncf.io/wp-content/uploads/2022/05/Q3-2021-State-of-Cloud-Native-development_FINAL.pdf, zuletzt abgerufen am 11.04.2023.

SLASHDATA (2022)
SlashData: State of the Developer Nation – 22nd Edition – The latest trends from our Q1 2022 survey of 20000 developers. Veröffentlicht im Jahr2022. Online-Quelle verfügbar unter https://slashdata-website-cms.s3.amazonaws.com/sample_reports/VZtJWxZw5Q9NDSAQ.pdf, zuletzt abgerufen am 11.04.2023.

SOJKA (2009)
Sojka, P.: An Experience with Building Digital Open Access Repository DML-CZ. In: CASLIN 2009 – 16th International Seminar – Institutional Online Repositories and Open Access, Teplá Monastery, Czech Republic, 7-11 June 2009. Pilsen 2009, S. 74-78.

STAAB (2011)
Staab, S.: Ontologies and Similarity. In: Ram, A.; Wiratunga, N. (Hrsg.): Case-based reasoning research and development – 19th International Conference on Case-Based Reasoning, ICCBR 2011, London, UK, September 12 - 15, 2011; Proceedings. Berlin 2011, S. 11-16.

STACK OVERFLOW (2021)
Stack Overflow: Stack Overflow Developer Survey 2021. Veröffentlicht im Jahr 2011. Online-Quelle verfügbar unter https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language, zuletzt abgerufen am 11.04.2023.

STACK OVERFLOW (2023)
Stack Overflow: Newest owlready Questions. Copyright 2023. Online-Quelle verfügbar unter https://stackoverflow.com/questions/tagged/owlready, zuletzt abgerufen am 11.04.2023.

STEYER (2018)
Steyer, R.: Programmierung in Python – Ein kompakter Einstieg für die Praxis, Wiesbaden 2018.

SYCHEV/ANIKIN/DENISOV (2021)
Sychev, O. A.; Anikin, A.; Denisov, M.: Inference Engines Performance in Reasoning Tasks for Intelligent Tutoring Systems. In: Gervasi, O.: Computational Science and Its Applications - ICCSA 2021 – 21st International Conference, Cagliari, Italy, September 13-16, 2021, Proceedings, Part II. Cham 2021, S. 471-482.

SYNERGY RESEARCH GROUP (2022)
Synergy Research Group: Q2 Cloud Market Grows by 29% Despite Strong Currency Headwinds; Amazon Increases its Share. Veröffentlicht im Jahr 2022. Online-Quelle verfügbar unter https://www.srgresearch.com/articles/q2-cloud-market-grows-by-29-despite-strong-currency-headwinds-amazon-increases-its-share, zuletzt abgerufen am 11.04.2023.

SYSKA (2006)
Syska, A.: Produktionsmanagement – Das A - Z wichtiger Methoden und Konzepte für die Produktion von heute. Wiesbaden 2006.

TABINDA KOKAB/ASGHAR/NAZ (2022)
Tabinda Kokab, S.; Asghar, S.; Naz, S.: Transformer-based deep learning models for the sentiment analysis of social media data. In: Array, Vol. 14 (2022), Article 100157, o. S. (beitragsindividuelle Paginierung S. 1-12).

TAYLOR/DU PREEZ (2023)
Taylor, R. M. C.; Du Preez, J. A.: SimLDA – A Tool for Topic Model Evaluation. In: Arai, K. (Hrsg.): Proceedings of the Future Technologies Conference (FTC) 2022. Cham 2023, S. 534-554.

TENSORFLOW (2023)
TensorFlow: Neural machine translation with a Transformer and Keras. Copyright 2023. Online-Quelle verfügbar unter https://www.tensorflow.org/text/tutorials/transformer, zuletzt abgerufen am 11.04.2023.

TENZER (2022)
Tenzer, F.: Anteil der privaten Haushalte in Deutschland mit einem Mobiltelefon von 2000 bis 2022. Veröffentlicht am 03.11.2022. Online verfügbar unter https://de.statista.com/statistik/daten/studie/198642/umfrage/anteil-der-haushalte-in-deutschland-mit-einem-mobiltelefon-seit-2000/, zuletzt abgerufen am 11.04.2023.

URBAN/GARLOFF (2022)
Urban, M.; Garloff, K.: Sovereign Cloud Stack. In: Datenschutz und Datensicherheit, Vol. 46 (2022), Issue 10, S. 616-621.

VASWANI et al. (2017)
Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I.: Attention Is All You Need. Veröffentlicht am 12.06.2017. Online-Quelle verfügbar unter https://arxiv.org/abs/1706.03762, zuletzt abgerufen am 05.05.2023.

VETTOR/SMITH (2023)
Vettor, R.; Smith, S.: Architecting Cloud-Native .NET Apps for Azure. Copyright 2023. Online-Quelle verfügbar unter https://learn.microsoft.com/pdf?url=https%3A%2F%2Flearn.microsoft.com%2Fen-us%2Fdotnet%2Farchitecture%2Fcloud-native%2Ftoc.json, zuletzt abgerufen am 18.04.2023.

VOIGT/VON DEM BUSSCHE (2018)
Voigt, P.; von dem Bussche, A.: EU-Datenschutz-Grundverordnung (DSGVO). Berlin – Heidelberg 2018.

WANG et al. (2022)
Wang, J.; Zhao, C.; He, S.; Gu, Y.; Alfarraj, O.; Abugabah, A.: LogUAD: Log Unsupervised Anomaly Detection Based on Word2Vec. In: Computer Systems Science and Engineering, Vol. 41 (2022), No. 3, S. 1207-1222.

WANG/LIN/ZHANG (2022)
Wang, H.; Lin, Q.; Zhang, Y.: Risk Cost Measurement of Value for Money Evaluation Based on Case-Based Reasoning and Ontology – A Case Study of the Urban Rail Transit Public-Private Partnership Projects in China. In: Sustainability, Vol. 14 (2022), S. 1-22.

WATSON (1998)
Watson, I. D.: Applying case-based reasoning – Techniques for enterprise systems. San Francisco 1998.

WATSON (2003)
Watson, I.: Applying Knowledge Management – Techniques for Building Corporate Memories. San Francisco 2003.

WEBER et al. (2021)
Weber, L.; Heeb, T.; Sethupathy, G.; Schagen, J. P.; Zelewski, S.: „Intelligente“ Wiederverwendung von Erfahrungswissen im betrieblichen Projektmanagement mithilfe von KI-Techniken bei sicherheitskritischen IT-Projekten mit Fokus auf PRINCE2 und Risikomanagement. Arbeitsbericht Nr. 50, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), zugleich KI-LiveS-Projektbericht Nr. 4. Essen 2021.

WEBER et al. (2023)
Weber, L.; Sethupathy, G.; Schagen, J. P.; Zelewski, S.: Design des User Interfaces für ein ontologiegestütztes Case-based-Reasoning-System zur „intelligenten“ Wiederverwendung von Erfahrungswissen – eine betriebswirtschaftliche Analyse im Hinblick auf sicherheitskritische IT-Projekte. Arbeitsbericht Nr. 65, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), zugleich KI-LiveS-Projektbericht Nr. 18. Essen 2023.

WILKE/BERGMANN (1998)
Wilke, W.; Bergmann, R.: Techniques and Knowledge used for Adaptation during Case-Based Problem Solving. In: del Pobil, A. P.; Mira, J.; Ali, M. (Hrsg.): Tasks and Methods in Applied Artificial Intelligence – 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA-98-AIE Benicàssim, Castellón, Spain, June 1-4, 1998, Proceedings, Volume II. Berlin – Heidelberg, S. 497-506.

WOHLGENANNT/MINIC (2016)
Wohlgenannt, G.; Minic, F.: Using word2vec to Build a Simple Ontology Learning System. Veröffentlicht im Jahr 2016. Online-Quelle verfügbar unter :http://ceur-ws.org/Vol-1690/paper 37.pdf, zuletzt abgerufen am 05.05.2023.

WORTH (2023)
Worth, P. J.: Word Embeddings and Semantic Spaces in Natural Language Processing. In: International Journal of Intelligence Science, Vol. 13 (2023), S. 1-21.

WYNER (2008)
Wyner, A.: An ontology in OWL for legal case-based reasoning. In: Artificial Intelligence and Law, Vol. 16 (2008), S. 361-387.

YOUN/NARAVANE/TAGKOPOULOS (2020)
Youn, J.; Naravane, T.; Tagkopoulos, I.: Using Word Embeddings to Learn a Better Food Ontology. In: Frontiers in Artificial Intelligence, Vol. 3 (2020), Article 584784, S. 584-784 (hier: beitragsindividuelle Paginierung S. 1-8).

ZELEWSKI (2005)
Zelewski, S.: Einführung in das Themenfeld „Ontologien“ aus informations- und betriebswirtschaftlicher Perspektive. In: Zelewski, S.; Alan, Y.; Alparslan, A.; Dittmann, L.; Weichelt, T. (Hrsg.): Ontologiebasierte Kompetenzmanagementsysteme – Grundlagen, Konzepte, Anwendungen. Berlin 2005, S. 115-228.

ZELEWSKI/HEEB/SCHAGEN (2022)
Zelewski, S.; Heeb, T.; Schagen, J. P.: Case-based Reasoning als White-Box AI: „intelligentes“ Projektmanagement durch die computergestützte Wiederverwendung von Erfahrungswissen in der betrieblichen Praxis – Teil 2: Das KI-Tool jCORA für ontologiegestütztes Case-based Reasoning im Projektmanagement. In: Bodemann, M.; Fellner, W.; Just, V. (Hrsg.): Digitalisierung und Nachhaltigkeit – Transformation von Geschäftsmodellen und Unternehmenspraxis. Berlin – Heidelberg 2022, S. 225-263.

ZELEWSKI/KOWALSKI/BERGENRODT (2015a)
Zelewski, S.; Kowalski, M.; Bergenrodt, D.: Intelligente Wissenswiederverwendung in internationalen Logistik-Projekten. In: Ege, B.; Humm, B.; Reibold, A. (Hrsg.): Corporate Semantic Web. Berlin – Heidelberg 2015, S. 289-305.

ZELEWSKI/KOWALSKI/BERGENRODT (2015b)
Zelewski, S.; Kowalski, M.; Bergenrodt, D.: Management von Erfahrungswissen aus internationalen Logistik-Projekten mithilfe von Case-based Reasoning. In: Zelewski, S.; Akca, N.; Kowalski, M. (Hrsg.): Organisatorische Innovationen mit Good Governance und Semantic Knowledge Management in Logistik-Netzwerken. Berlin 2015, S. 229-267.

ZELEWSKI/SCHAGEN (2022)
Zelewski, S.; Schagen, J. P.: Case-based Reasoning als KI-Technik zur „intelligenten“, computergestützten Wiederverwendung von Erfahrungswissen im Projektmanagement. Arbeitsbericht Nr. 55, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), zugleich KI-LiveS-Projektbericht Nr. 9. Essen 2022.

ZHANG et al. (2023)
Zhang, S.; Fan, R.; Liu, Y.; Chen, S.; Liu, Q.; Zeng, W.: Applications of transformer-based language models in bioinformatics: a survey. In: Bioinformatics advances, Vol. 3 (2023), Issue 1, S. 1-19.

ZIMA (2015)
Zima, K.: The Case-based Reasoning Model of Cost Estimation at the Preliminary Stage of a Construction Project. In: Procedia Engineering, Vol. 122 (2015), S. 57-64.

Herr Dr. **Ganen Sethupathy** studierte Geoinformatik (M.Sc.) und leitet derzeit einen Bereich in einem IT-Beratungskonzern. Parallel zu seiner beruflichen Tätigkeit absolvierte er ein MBA-Studium, das ihn unter anderem nach New York und San Diego führte. Im Anschluss promovierte er berufsbegleitend an der Universität Duisburg-Essen im Rahmen des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten Forschungs- und Transferprojekts „KI-LiveS“ über betriebliche Anwendungen der KI-Technik „ontologiegestütztes Case-based Reasoning“. Seine Forschungsschwerpunkte liegen in den Bereichen Ontologien, Case-based Reasoning und Künstliche Neuronale Netze. Darüber hinaus vertieft er seine Expertise im Bereich Künstliche Intelligenz am Massachusetts Institute of Technology (MIT) durch die Teilnahme an einem dreijährigen Programm zu Machine Learning and Artificial Intelligence. Herr Dr. Sethupathy engagiert sich zudem besonders für die Digitalisierung der öffentlichen Verwaltung mit einem besonderen Augenmerk auf den Schutz und die Implementierung kritischer Infrastrukturen.

Herr **Jan Peter Schagen**, M.Sc., ist wissenschaftlicher Mitarbeiter und Promotionsstudent am Institut für Produktion und Industrielles Informationsmanagement an der Fakultät für Wirtschaftswissen-schaften der Universität Duisburg-Essen. Er studierte an der Universität Duisburg-Essen, Campus Essen, den Bachelorstudiengang Betriebswirtschaftslehre (B.Sc.) und anschließend den Masterstudiengang Märkte und Unternehmen (M.Sc.). Im Rahmen seiner Tätigkeit als wissenschaftlicher Mitarbeiter arbeitete er unter anderem im Verbundprojekt KI-LiveS, das anstrebte, Techniken aus der Erforschung Künstlicher Intelligenz in die betriebliche Praxis zu transferieren. Sein Dissertationsprojekt konzentriert sich auf die „intelligente“ Wiederverwendung von Erfahrungswissen im Projektmanagement mittels ontologiegestützten Case-based Reasonings mit einem besonderen Fokus auf die Konzipierung, Implementierung und Evaluierung von Adaptionsregeln für ein ontologiegestütztes Case-based-Reasoning-System.

Herr Univ.-Prof. Dr. **Stephan Zelewski** ist Inhaber einer Professur für Betriebswirtschaftslehre und Direktor des Instituts für Produktion und Industrielles Informationsmanagement an der Fakultät für Wirtschaftswissenschaften der Universität Duisburg-Essen. Er studierte Betriebs- und Volkswirtschaftslehre (Dipl.-Kfm., Dipl.-Volksw.) an den Universitäten Münster sowie Köln. Im Jahr 1985 wurde er an der Universität zu Köln mit einer Arbeit über betriebswirtschaftliche Anwendungspotenziale der Künstlichen Intelligenz promoviert. Nach der Habilitation zur Strukturalistischen Produktionstheorie war er in den Jahren 1993 bis 1998 Inhaber einer Professur für Betriebswirtschaftslehre sowie Direktor des Instituts für Produktionswirtschaft und Industrielle Informationswirtschaft an der Universität Leipzig. Die Hauptarbeitsgebiete von Herrn Zelewski sind einerseits Fragestellungen des computergestützten Produktionsmanagements an der Nahtstelle zwischen Betriebswirtschaftslehre und Wirtschaftsinformatik mit Schwerpunkten in den Bereichen Projektmanagement, Logistik, Supply Chain Management sowie Produktionsplanung und -steuerung. Andererseits befasst er sich intensiv mit dem Transfer von Erkenntnissen aus der Erforschung Künstlicher Intelligenz auf ökonomische Probleme, insbesondere im Hinblick auf Wissensbasierte Systeme, Ontologien und Case-based Reasoning.

In sicherheitskritischen IT-Projekten, deren Erfolg entscheidend für die öffentliche Sicherheit ist, kann der Ausfall eines IT-Systems weitreichende Folgen haben. Die Wiederverwendung von wertvollem Erfahrungswissen, das oft in natürlicher Sprache und auf unterschiedlichen IT-Systemen oder in physischen Dokumenten vorliegt, stellt in der Praxis jedoch eine erhebliche Herausforderung dar. Wird dieses Erfahrungswissen nicht berücksichtigt, können die Konsequenzen gravierend sein und zu erheblichen Mehrkosten sowie Projektverzögerungen führen.

Ein möglicher Ansatz zur Lösung dieser Probleme liegt an der Schnittstelle von Betriebswirtschaftslehre, Wirtschaftsinformatik und Informatik. Dieser Ansatz vereint die Disziplinen Projektmanagement, Wissensmanagement und Künstliche Intelligenz (KI). Er zeigt, wie „traditionelle“ KI-Techniken, vor allem Ontologien und Case-based Reasoning, in sicherheitskritischen IT-Projekten effektiv eingesetzt werden können. Durch die Kombination von ontologiegestütztem Case-based Reasoning mit Künstlichen Neuronalen Netzen für die „moderne“ Word2Vec-Technik wird eine Cloud-basierte Software konzipiert, mit deren Hilfe sich die Wiederverwendung von Erfahrungswissen in sicherheitskritischen IT-Projekten deutlich verbessern lässt.

Dieser Beitrag zeigt auf, wie die Verbindung von traditionellen und modernen KI-Techniken einen Schlüssel zur systematischen Wiederverwendung von erfolgskritischem Erfahrungswissen in sicherheitskritischen IT-Projekten bieten und einen wichtigen Beitrag zur öffentlichen Sicherheit leisten kann.

Logos Verlag Berlin **ISBN 978-3-8325-5883-3**