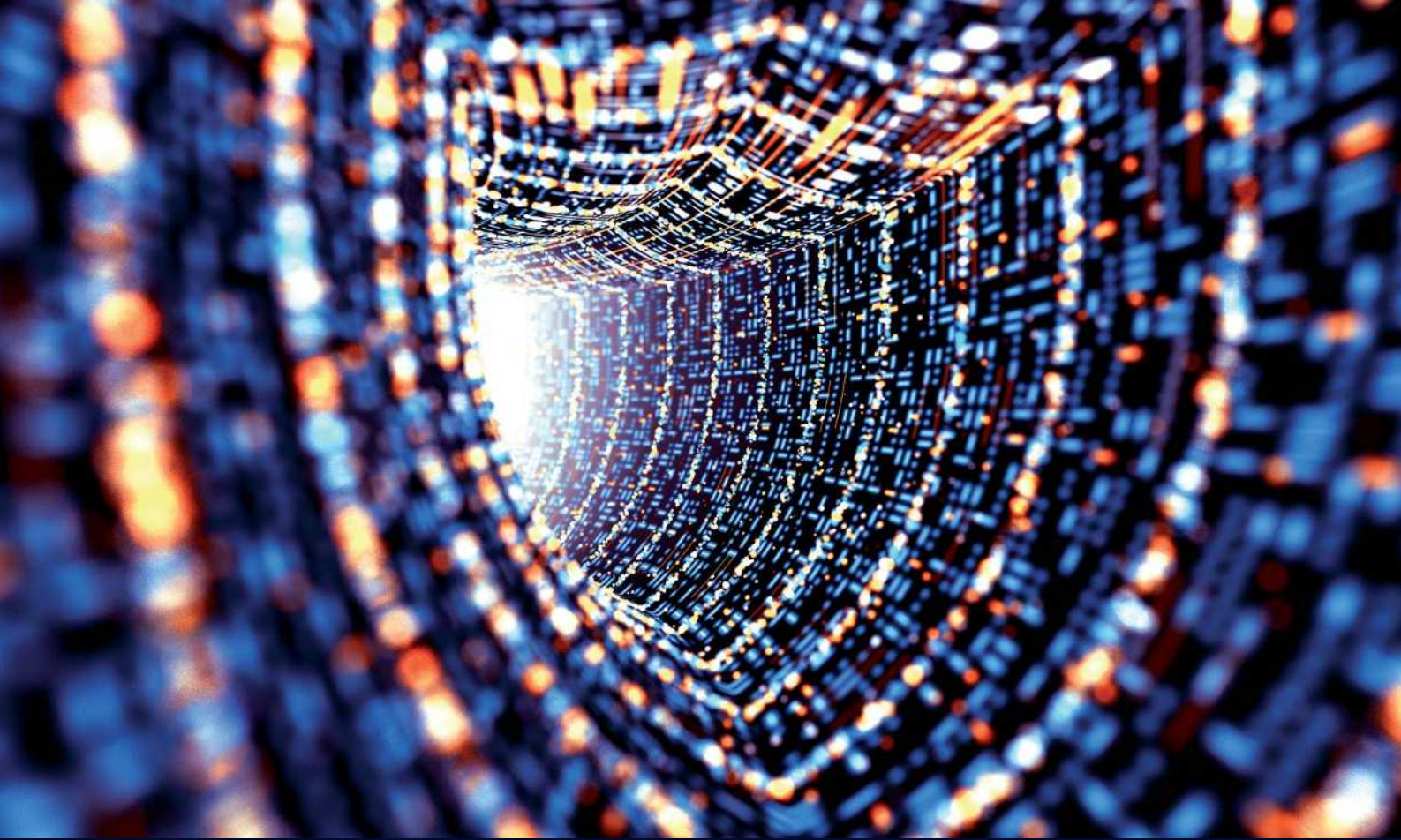




Ganen Sethupathy
Jan Schagen
Stephan Zelewski

Knowledge-based Project Management with the Help of AI and Cloud Techniques

**Systematic Reuse of Experience-based Knowledge
about Safety-critical IT Projects Based on
Ontologies, Case-based Reasoning and Word2Vec**

λoγoς

Ganen Sethupathy, Jan Peter Schagen, Stephan Zelewski

Knowledge-based Project Management with the Help of AI and Cloud Techniques

Systematic Reuse of Experience-based Knowledge
about Safety-critical IT Projects Based on
Ontologies, Case-based Reasoning and Word2Vec

Logos Verlag Berlin



Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche
Nationalbibliografie; detailed bibliographic data are available in the
Internet at <http://dnb.d-nb.de>.

Cover image: iStock, BlackJack3D

This work is licensed under the Creative Commons license CC BY-NC-ND
(<https://creativecommons.org/licenses/by-nc-nd/4.0/>). Creative Commons license
terms for re-use do not apply to any content (such as graphs, figures, photos,
excerpts, etc.) not original to the Open Access publication and further permission
may be required from the rights holder. The obligation to research and clear
permission lies solely with the party re-using the material.



Logos Verlag Berlin GmbH 2025
ISBN 978-3-8325-5902-1
DOI: <https://doi.org/10.30819/5902>

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.com>

Table of Contents

1	Introduction to the reuse of experience-based knowledge about safety-critical IT projects	1
1.1	Real-world problems in the reuse of experience-based knowledge about safety-critical IT projects	1
1.2	Business desiderata for the reuse of experience-based knowledge about safety-critical IT projects	4
1.3	State of the art of the techniques available to fulfill the business desiderata.....	5
1.4	Scientific problems with regard to the reuse of experience-based knowledge about safety-critical IT projects	8
1.5	Intended scientific findings	10
2	Foundations for ontology-supported case-based reasoning for the reuse of experience-based knowledge about safety-critical IT projects.....	13
2.1	Ontologies	13
2.1.1	Classification of the concept of ontology in an information and business management context	13
2.1.2	Definition of ontologies	14
2.1.3	Ontology components	14
2.1.4	Ontology-related representation languages.....	17
2.2	Case-based reasoning	18
2.2.1	Basic idea of case-based reasoning.....	18
2.2.2	Case-based reasoning cycle	19
2.3	Project management of security-critical IT projects	20
2.3.1	Project management.....	20
2.3.2	Project management domain: Security-critical IT projects	22
2.4	IT applications.....	24
2.4.1	Monolithic applications	24
2.4.2	Cloud-native applications	24

3	Application of ontology-supported case-based reasoning for the reuse of empirical knowledge about safety-critical IT projects	27
3.1	Procedure for the application of ontology-supported case-based reasoning for the reuse of empirical knowledge about safety-critical IT projects	27
3.2	Construction of a security-critical IT project ontology	27
3.2.1	Selection of a design method for the development of a safety-critical IT project ontology	27
3.2.2	Selecting Protégé as an ontology editor	28
3.2.3	Application of the construction method for the construction of a safety-critical IT project ontology	30
3.2.3.1	Defining the scope of an IT project ontology	30
3.2.3.2	Testing existing ontologies for their use in the construction of a new IT project ontology	34
3.2.3.3	Definition of important terms for the IT project ontology	37
3.2.3.4	Class construction	39
3.2.3.5	Construction of non-taxonomic relations	90
3.2.3.6	Constructing attributes	97
3.2.3.7	Construction of cardinalities	104
3.2.3.8	Construction of Semantic Web Rules	109
3.2.3.9	Construction of global individuals	118
3.3	Case-based reasoning on the basis of a safety-critical IT project ontology	129
3.3.1	Ontology-supported case-based reasoning system using jCORA	129
3.3.1.1	Description of the prototype CBR tool jCORA	129
3.3.1.2	Using the CBR tool jCORA for case specification	132
3.3.1.3	Use of the CBR tool jCORA for case-related similarity calculation	142
3.3.1.4	Limitations of the CBR tool jCORA	146
3.3.2	Description of the cases to represent three practical examples	156
3.3.2.1	Preliminary remarks on the three practical examples	156
3.3.2.2	Case 1: Reorganization of a police command and control system	158
3.3.2.3	Case 2: Setting up a cooperative control center	162
3.3.2.4	Case 3: Creation of a central database for investigations	166
3.3.3	Similarity calculations	170
3.3.3.1	Calculation basis of the CBR tool jCORA	170
3.3.3.2	Exemplary similarity calculation using the CBR tool jCORA	177

4	Design of an ontology-supported case-based reasoning system as a cloud-native application	181
4.1	Preliminary remarks on the system design.....	181
4.2	Cloud environments	182
4.3	Designing an ontology-supported case-based reasoning system as a cloud-native application.....	188
4.3.1	Designing the frontend.....	188
4.3.2	Design of the backend.....	195
4.3.2.1	Preliminary considerations for backend design.....	195
4.3.2.2	Designing the middleware for the backend of a cloud-native application.....	200
5	Critical reflections	229
5.1	Ontology-supported case-based reasoning for the reuse of experience-based knowledge about safety-critical IT projects.....	229
5.2	Cloud-native application for an ontology-supported case-based reasoning system.....	232
6	Conclusions on the scientific findings obtained	239
7	Outlook for further research needs	243
	Bibliography	247

1 Introduction to the reuse of experience-based knowledge about safety-critical IT projects

1.1 Real-world problems in the reuse of experience-based knowledge about safety-critical IT projects

The studies¹ presented here deal with the great importance of security-critical IT systems for companies in the commercial sector, public institutions, and the civilian population. This importance has risen sharply in recent years, as we will briefly explain using two concise examples.

A security-critical IT system is one considered relevant to the security of the civilian population and whose potential failure is associated with a threat to public security. Safety-critical IT systems include critical infrastructures based on information technology, such as a command and control system or the digital radio of authorities and organizations with security tasks. Security-critical IT systems fall into the area of critical infrastructures (KRITIS). They are defined as “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), p. 3).

In addition to economic requirements, safety-critical IT systems are also subject to social ones. The failure or delayed commissioning of a safety-critical IT system can endanger human lives, have serious economic consequences, and lastingly impact the civilian population’s security needs. Safety-critical IT systems are provided through safety-critical IT projects. The reuse of existing knowledge—so-called experience-based knowledge—is a critical success factor for the successful implementation of safety-critical IT projects. Problems caused by a lack of experience in safety-critical IT projects can lead to the failure of said projects, or to a lack of added project value. The following two practical examples from Europe illustrate this.

1) This article is essentially based on the dissertation of the first-named author; cf. SETHUPATHY (2024). Revisions—in particular a significant streamlining of the content and the English-language translation—were made by the second- and third-named authors (and by a professional translation agency on their behalf). The underlying (German-language) publication SETHUPATHY (2024) contains extensive additional details, a very detailed appendix—especially with extensive information on the ontology and the CBR system—as well as a wealth of in-depth references. As is customary in “modern” international publications, we have deliberately made very sparing use of footnote references so as not to interrupt the reading flow and the layout’s visual appearance too much. Readers who are interested in much more detailed source references, including the associated source criticism, are asked to consult the first-named author’s dissertation—cf. SETHUPATHY (2024)—directly.

The safety-critical IT project “Kooperative Leitstelle Berlin” which is currently in the implementation phase, has increased in cost from the original 84 million euros to around 250 million euros due to planning errors and a delay in awarding the contract. As an additional consequence, the project is expected to be delayed by seven years. As the maintenance of the current control centers can only be guaranteed until 2025, the bridging period will necessitate an alternative solution, further raising costs as well as jeopardizing society’s sense of security with regard to emergency call handling.

Another example is the renewal of the Swiss Polycom security radio network, where underestimated requirements and deficiencies produced project delays. Here, too, the critical scheduling and the associated possible need for parallel operation of the new and old systems have raised the likelihood of additional costs. As a result of this and the direct awarding of the contract without a public tender, the project is the subject of public discussion with the risk of growing mistrust in society. The client publicly announced that there was a lack of sound and specialized knowledge. A press release on the matter states: “Dieses Wissen aufzubauen hat sich aufseiten des Auftragnehmers als komplexer und schwieriger als ursprünglich angenommen herausgestellt.” (DER BUNDESRAT DER SCHWEIZER REGIERUNG (2021).

These practical examples illustrate that the use of expertise and experience is a critical success factor for security-critical IT projects. In this context, one can also speak of “project knowledge management”, because project management is a particularly knowledge-intensive management task. The close connection between knowledge and project management is also emphasized in various publications and by common project management standards (such as PRINCE2, PMI, Scrum, and IPMA). The standards recommend drawing experience from completed projects, which is documented, for example, in lessons learned, sprint retrospectives, and phase completion reports. This experience should be utilized in similar projects or in further sprints in order to enable engagement with positive and negative experiences.

The relevance of the reuse of knowledge on the one hand and its inadequate consideration in everyday operations on the other illustrates the problem-oriented area of tension in the reuse of knowledge in project management in general and in the project management of safety-critical IT projects in particular. This *reuse problem* comprises four sub-problems, which we briefly list and then explain in terms of their content and further sub-summarized sub-problems.

The first problem to be considered is that of *knowledge representation*. Knowledge gained from experience in operational project management is usually stored in the form of documents, plans, or logs. However, this form of storage is problematic for several reasons:

- Structuring problem: The documents are usually available in an unstructured form.
- Source problem: In a company environment, there are also different sources that can be used to store such documents, which is why there is no central storage location for the storage of experience-based knowledge.
- Vocabulary problem: The documentation of the experiential knowledge of different actors is heterogeneous, so that the documents lack a uniformly defined vocabulary that is understood in the same way by all actors.

The *system problem* is reflected in the lack of a suitable IT system for the reutilization of experience-based knowledge from safety-critical IT projects. The lack of suitability can be categorized in problem-oriented terms as follows:

- Processing problem: Existing IT systems have difficulty processing unstructured knowledge.
- Accessibility problem: For an IT system to be accepted, it is also necessary that experience-based knowledge can be accessed at any time and in any place. This is currently not possible.
- Intelligence problem: As a rule, existing IT systems are limited to simply displaying the documents. There exists no “intelligent” processing of the document content.

The *knowledge loss problem* comprises the loss of experiential knowledge. This can occur for various reasons, which are subsumed under the loss of knowledge problem in problem-orientated diction as follows:

- Project team turnover problem: Constantly reorganized project teams create dynamism, but lead to the loss of specialist knowledge and experience.
- Dismissal problem: Planned or unplanned company departures can lead to a loss of knowledge.

The *knowledge evaluation problem* comprises all problems associated with the evaluation of existing experience-based knowledge:

- Resource problem: In most IT systems, the search for experience-based knowledge takes too long and is too resource-intensive, as those researching must comb through heterogeneous sources and natural-language texts.

- Technical problem: With existing techniques, natural language knowledge components can be identified and automatically analyzed only to an insufficient degree.
- Comparison problem: The comparability of safety-critical IT projects is problematic, as no standard exists for the systematic comparison of safety-critical IT projects.

1.2 Business desiderata for the reuse of experience-based knowledge about safety-critical IT projects

The business desiderata can be derived on the basis of the previous explanations. The overarching desideratum, referred to below as the reutilization desideratum, is the creation of a way of intelligently reusing experience gained from safety-critical IT projects. From this, we derive the following partial desiderata:

- Knowledge representation desideratum
- System desideratum
- Knowledge evaluation desideratum
- Knowledge loss desideratum

The *knowledge representation desideratum* expresses the need for a comprehensive representation of experiential knowledge and comprises the following sub-desiderata:

- Structuring desideratum: It would be desirable to structure experience-based knowledge.
- Source desideratum: It would be desirable to store experience-based knowledge centrally at a single source.
- Vocabulary desideratum: It would be desirable to represent experiential knowledge using a standardized vocabulary.

The *system desideratum* expresses that it would be desirable to provide an IT system that enables the reuse of experience-based knowledge. The system desideratum comprises the following sub-desiderata:

- Processing desideratum: It would be desirable to be able to process unstructured knowledge.
- Accessibility desideratum: It would be desirable to be able to make experiential knowledge accessible anywhere and at any time.

- Intelligence desideratum: It would be desirable to reuse experience-based knowledge “intelligently”.

The *knowledge loss desideratum* expresses that it would be desirable to store all knowledge gained from experience in order to prevent a loss of knowledge. The knowledge loss desideratum comprises the following subdesiderata:

- Project team change desideratum: It would be desirable to be able to store the experience of departing project employees when project teams change.
- Resignation desideratum: It would be desirable to be able to store the experience of departing employees.

The *knowledge evaluation desideratum* expresses the need for a systematic evaluation of stored experience-based knowledge and comprises the following subdesiderata:

- Resource desideratum: It would be desirable to be able to carry out the evaluation of experiential knowledge effectively and efficiently.
- Technology desideratum: It would be desirable to include techniques that can also evaluate natural language knowledge components.
- Comparison desideratum: It would be desirable to have systematic benchmarks for the evaluation of safety-critical IT projects.

1.3 State of the art of the techniques available to fulfill the business desiderata

An examination of the state of the art shows that the combination of ontologies and case-based reasoning (CBR) is fundamentally suitable for the intelligent reuse of experiential knowledge in project management. Ontology-supported case-based reasoning combines the two techniques of ontologies and case-based reasoning. The use of ontologies in case-based reasoning systems (CBR systems for short) is not new. It has already been dealt with in various research publications; cf. DUARTE/BELO (2023), pp. 830–842; NKISI-ORJI et al. (2022), pp. 127–138; OBEID et al. (2022), pp. 991–1002; WANG/LIN/ZHANG (2022), pp. 4–19; WEBER et al. (2021), pp. 12–27; EMMENEGGER et al. (2017), pp. 338–351; MARTIN et al. (2017), pp. 552–571; BOUHANA et al. (2015), pp. 3726–3740; ZELEWSKI/KOWALSKI/BERGENRODT (2015a), pp. 294–302; ZELEWSKI/KOWALSKI/BERGENRODT (2015b), pp. 242–255; RECIO-GARCÍA/GONZÁLEZ-CALERO/DÍAZ-AGUDO (2014), pp. 129, 134, and 137–143; AMAILEF/LU (2013), pp. 81–96; GUO/HU/PENG (2012), pp. 497–507; BEIBEL (2011), pp. 40–220; ASSALI/LENNE/DEBRAY (2010), pp. 97–115; WYNER (2008), pp. 361–385; RECIO-GARCÍA et al. (2007), pp. 151–

161; BERGMANN/SCHAAF (2003), pp. 608–624. This use of ontologies in CBR systems is generally regarded as positive; cf. WANG/LIN/ZHANG (2022), p. 5; BEIBEL (2011), p. 40. The advantage of combining ontologies with CBR systems is primarily seen in the fact that an ontology specifies the standardized vocabulary that is used in a CBR system. More precisely, this refers to the linguistic means of expression that can be used in an ontology-supported CBR system to specify and solve problems. These linguistic means of expression can go beyond a simple vocabulary, such as non-taxonomic relations or inference and integrity rules—as we will discuss later in more detail.

We first describe the two techniques separately, then explain the combination of ontology-supported case-based reasoning.

Ontologies make it possible to create a shared understanding of the linguistic means of expression used in particular “conceptualizations” (in addition to, e.g., inter-conceptual semantic relations and semantic axioms with regard to the correct use of terms as to content). These linguistic means of expression are conceptualized and formally defined in ontologies; cf. CAROLLA (2015), p. 31.

Ontologies that relate to the domain of project management can already be found in the state of the art; cf. SANTOS JÚNIOR et al. (2021), pp. 7–25, with a focus on agile projects; WEBER et al. (2021), pp. 12–23 and 50–75, with a focus on security-critical IT projects; MARTIN et al. (2017), pp. 551–552, 562, and 567–570; ZELEWSKI/KOWALSKI/BERGENRODT (2015b), pp. 245–250; LIN et al. (2012), pp. 195–206; SHEEBA/KRISHNAN/BERNARD (2012), pp. 2–7; DONG/HUSSAIN/CHANG (2011), pp. 1164–1169; HUGHES (2010), pp. 9–19; ARAMO-IMMONEN (2009), pp. 49–55; SARANTIS/ASKOUNIS (2009), pp. 2–7; ABELS et al. (2006), pp. 817–819. However, researchers are not yet focusing on the domain of safety-critical IT projects separately. Only WEBER et al. (2021) have made an initial attempt to conceptualize a project management ontology for safety-critical IT projects.

Case-based reasoning rests on the fundamental idea of comparing new cases with those old cases that have already been solved, then applying the most suitable old case to the new case. In the field of project management, this means that the knowledge gained from previous, similar projects should be reused for the implementation of new projects. However, the current fields of application of case-based reasoning as an isolated method for project management are limited to quantitative knowledge components, such as for estimating project costs; cf. for example RADZIEJOWSKA/ZIMA (2015), pp. 100–111; ZIMA (2015), pp. 59–64; KIM/SHIM (2014), pp. 66–72; JI/PARK/LEE (2012), pp. 45–51; KIM et al. (2012), pp. 284–291. Project-specific—especially qualitative (natural language)—experience, such as requirement and functional specifications or employee competencies, are rarely or only insufficiently taken into account.

The combination of the two AI techniques “ontologies” and “case-based reasoning”—ontology-supported case-based reasoning—can be used promisingly for knowledge management in project management. Some CBR tools already exist for the construction of ontology-supported CBR systems for its implementation. Examples include the CBR tools jCOLIBRI, MyCBR, COBRA, CASBIAN, CBR-Shell, Induce-IT, and KAI-DARA Advisor; cf., e.g., MARTIN et al. (2017), pp. 556–557.

jCORA (java-based Case- and Ontology-based Reasoning Application) is a CBR tool developed specifically for the application of ontology-based case-based reasoning in project management. WEBER et al. (2021), pp. 23–43, specifically discuss this tool’s use for safety-critical IT projects. However, the project ontology is not sufficiently expressive to adequately represent safety-critical IT projects from an operational perspective. In particular, the conceptualization of WEBER et al.’s publication was not based on performance descriptions and announcements from safety-critical IT projects. Furthermore, the ontology does not define its terms in accordance with the PRINCE2 standard.

A current research project on a CBR system based on cloud technologies is the project CLOOD; cf. NKISI-ORJI et al. (2022), pp. 125–138; NKISI-ORJI et al. (2020), pp. 132–142. This research project provides the CBR cycle’s phases as independent, publicly accessible, and usable serverless functions in the Python programming language, so that the functions can be used for any cloud-based CBR implementation project. However, there currently exists no implementation for cloud-based, ontology-supported case-based reasoning for the domain of project management or safety-critical IT projects.

Similarity measures for the processing of qualitative knowledge components are carried out in an ontology-supported CBR system by means of similarity algorithms, which particularly rely on the semantic distance within taxonomy graphs of an ontology to determine similarities. Alternatively, similarity tables are often used, but their concrete design is subject to rather subjective heuristics without a methodological background.

Artificial neural networks (ANN) offer starting points for determining the similarity of terms. Although NKISI-ORJI’s aforementioned publication offers an initial basis for the connection between ontology-based CBR systems and artificial neural networks, it remains vague with regard to an implementation proposal and the preparation of the training data.

1.4 Scientific problems with regard to the reuse of experience-based knowledge about safety-critical IT projects

In the following, we compare each business desideratum on the desideratum hierarchy's lowest level with the respective state of the art in order to check whether there is a non-trivial discrepancy in each case.

We begin by describing the non-trivial discrepancy between the sub-desiderata of the system desiderata and the state of the art for ontology-supported case-based reasoning.

- *Structuring desideratum*: By using an ontology in a CBR system, it is possible to store experience-based knowledge in a structured manner. However, no ontology exists that provides sufficient vocabulary for safety-critical IT projects.
- *Source desideratum*: In a CBR system, experience-based knowledge from safety-critical IT projects can be stored centrally. However, there currently exists no adequate implementation of a central ontology-supported CBR system for safety-critical IT projects.
- *Vocabulary desideratum*: By using an ontology, experience-based knowledge can be stored using a previously defined standardized vocabulary. However, there exists no ontology that provides a sufficient vocabulary for safety-critical IT projects.
- *Intelligence desideratum*: In principle, ontology-supported CBR systems are able to reuse experience-based knowledge intelligently. However, there exists no ontology for the domain of safety-critical IT projects in which the necessary linguistic means of expression have been specified to enable the intelligent reuse of experience-based knowledge in this domain. The CBR tools considered in the state of the art, in particular jCORa, can hardly be used in operational practice due to the limitations of the applied software technology and usability. It is evident that the CBR systems are perceived as inadequate in business practice.
- *Processing desideratum*: In principle, ontology-supported CBR systems support the processing of qualitative knowledge components, but there so far exists no fully comprehensive application for safety-critical IT projects, as there is no ontology for the domain of safety-critical IT projects in which the necessary linguistic means of expression have been specified.
- *Accessibility desideratum*: In principle, CBR tools for the construction of ontology-supported CBR systems, such as jCORa, can be accessed ubiquitously. However, ubiquitous use is subject to a number of application barriers, such as the need to install a Java environment and exclusive access via a desktop client.

In order to enable ubiquitous use, initial approaches exist to develop cloud-based CBR systems, and in some cases also ontology-supported CBR systems. However, these approaches cannot be regarded as mature systems that can be used in the context of security-critical IT projects.

- *Termination desideratum*: In principle, an ontology-supported CBR system enables the storage of experiential knowledge of employees leaving a company. For the domain of safety-critical IT projects, there exists no ontology in which the necessary linguistic means of expression have been modeled.
- *Project team change desideratum*: In principle, an ontology-supported CBR system makes it possible to store the experiential knowledge of employees leaving a project team. For the domain of safety-critical IT projects, there exists no ontology in which the necessary linguistic means of expression have been specified.
- *Resource desideratum*: In principle, it is possible to evaluate experience-based knowledge with CBR systems that have been constructed using standard CBR tools such as jCORA. However, the processing steps, such as the similarity calculation, cause a high local computational load, which cannot be adequately distributed across several computing resources due to the monolithic application structure. Cloud-native applications offer possible solutions for distributing the computational load that are lacking in conventional monolithic CBR tools. A possible implementation of ontology-supported case-based reasoning as a cloud-native application for safety-critical IT projects does not currently exist.
- *Technology desideratum*: Although the evaluation of natural language knowledge components is possible in principle using an ontology-based CBR system, the use of similarity tables in particular is not always effective and efficient in practice. Artificial neural networks, which can be used by cloud-native applications, offer starting points for the evaluation of natural language knowledge components on the basis of comparative measures.
- *Comparison desideratum*: CBR systems constructed with standard CBR tools such as jCORA offer universal benchmarks for comparing projects. Specific benchmarks for safety-critical IT projects do not currently exist. Due to the monolithic application structure, the similarity functions for the benchmarks are also tied to the CBR systems. Extensions, such as the implementation of additional specific similarity functions for safety-critical IT projects, require development in the CBR systems. Cloud-native applications do not have a monolithic application structure, which means that specific similarity functions can be provided

without system adaptation. However, there currently exist no ontology-based benchmarks that can be used for this purpose.

The above comparison reveals four significant non-trivial discrepancies:

- There exists no ontology that provides all relevant linguistic means of expression for security-critical IT projects.
- There exists no implementation of an ontology-based CBR system for safety-critical IT projects.
- There exists no ontology-supported case-based reasoning as a cloud-native application for safety-critical IT projects, including the implementation of ontology-supported benchmarks.
- There exist no benchmarks for ontology-based CBR systems for security-critical IT projects using artificial neural networks.

In problem-oriented diction, the aforementioned non-trivial discrepancies can be formulated as the following scientific problems:

- *Ontology problem*: There is a transfer problem in the transfer of the general technology of ontologies to the domain of safety-critical IT projects.
- *CBR system problem*: There is a transfer problem in transferring a general ontology-supported CBR system to the domain of safety-critical IT projects.
- *Cloud-native problem*: There is a transfer problem when transferring the general techniques of the cloud-native approach and ontology-supported, case-based reasoning with regard to the domain of safety-critical IT projects.
- *Neural network problem*: There is a problem of understanding how artificial neural networks can be used to create ontology-based CBR systems for safety-critical IT projects.

1.5. Intended scientific findings

We do not intend to implement a fully functional prototype as a cloud-native application. Instead, the prototypical development of the similarity algorithm and the implementation of specific similarity functions “only” serve to demonstrate the feasibility of an ontology-based CBR system as a cloud-native application. In summary, we intend to deliver the following scientific findings:

- Safety-critical IT project ontology
- CBR system with integrated ontology for safety-critical IT projects
- safety-critical IT projects in the form of projects in a CBR system
- Similarity calculation in the CBR system
- Similarity algorithm as a serverless function in a cloud environment
- Similarity functions as serverless functions in a cloud environment including specific similarity functions for the processing of qualitative information from safety-critical IT projects using artificial neural networks

2 Foundations for ontology-supported case-based reasoning for the reuse of experience-based knowledge about safety-critical IT projects

2.1 Ontologies

2.1.1 Classification of the concept of ontology in an information and business management context

The term “ontology” originally stems from philosophy, where it relates to the doctrine of existence. Philosophers define ontology as the study of the conceptual understanding, content analysis, and characterization of the interlocking of the most fundamental structures of existence. They see it an attempt to describe the basic structures of reality correctly and universally.

Yet this philosophical definition does not underpin our understanding of the term “ontology” that has gained relevance in the context of knowledge management. For this purpose, it is necessary to consider the concept of ontology from an information technology perspective.

In computer science, ontologies are linguistic means of expression for a common area of application—a “domain”. They are understood as a shared intellectual and linguistic understanding of this domain that supports communication between human individual and collective actors (persons, groups of persons, and companies) and artificial actors (computers, “machines”, software programs) and thus facilitates the exchange, sharing, and joint application of knowledge in companies. For these purposes, ontologies in the field of informatics provide linguistic means of expression for the representation of knowledge. In particular, they make it possible to incorporate the meaning of “qualitative” knowledge represented in natural language into computer-aided knowledge storage and processing.

The aim of an ontology is to create a shared conceptualization for the respective communication participants for a precisely specified section of reality (domain). Thus, an ontology helps document and make transparent the knowledge of relevant terms in a domain.

To summarize, the use of ontologies is expected to strengthen systematic knowledge management by structuring domain knowledge. To this end, relevant terms from a domain are specified in an ontology, taking into account their semantics and their relationships to each other. The representation of this knowledge is usually computer-aided through the use of ontology editors in a computer-understandable form using a computer-readable representation language.

2.1.2 Definition of ontologies

There currently exists no general definition of ontologies in computer science. One of the most frequently used definitions of ontologies comes from GRUBER, who defines an ontology as follows (GRUBER (1993), p. 199):

“An *ontology* is an explicit specification of a conceptualization.”

Although these and related definitions are very popular, they have also faced criticism from various authors, as the terms “conceptualization” and “explicit specification” in particular remain undefined. An additional point of criticism is that ontologies require that their construction be accompanied by a process of defining terms. This definition takes place with several actors as a shared, commonly accepted lexicon of terms, as the advantage of an ontology would be minimal no common understanding existed.

In order to avoid the points of criticism mentioned, we consider ZELEWSKI’s definition as more appropriate for this article. This is as follows (ZELEWSKI (2005), p. 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 defines ontologies in particular as a specification of *linguistic means of expression*. We apply his concept as a working definition for the following three reasons. The first lies in its language orientation. The language orientation is vital for the present investigations because we are using ontologies in the context of ontology-supported case-based reasoning for the provision of linguistic means of expression for the application of a CBR system. The second reason is that ZELEWSKI’s ontology definition formulates two further important aspects that GRUBER’s definition leaves out: Firstly, the section of reality to be considered on which perception and imagination are based must be identified. A further aspect concerns the conceptualization of a section of reality that is limited to relevant phenomena that are important for the actors’ communication.

2.1.3 Ontology components

Ontologies essentially consist of the following components: classes (or, understood here synonymously, concepts), relations, attributes, restrictions, inference rules, and integrity rules. However, it is not necessary to use all six of these when creating an ontology.

Classes represent terms for the linguistic structuring of a section of reality. A class comprises a set of individuals that are defined by common characteristic properties and are represented by a specific designation. Classes are organized in a taxonomic structure. A taxonomy organizes the relationships between the classes using the “is_a” relation. The advantage of a taxonomy is that inheritance mechanisms are applicable. A subclass inherits the properties of its superclass.

Classes can be further specified using relations and attributes. Relations and attributes are similar in their basic functions, which is why the specialist literature often summarizes them under the term “properties”. Relationships between classes can be established in both taxonomic and non-taxonomic form by means of relations. To establish taxonomic relationships between two classes, the taxonomic relation “is_a” is used. All other relations are referred to as non-taxonomic relations. Non-taxonomic relations can be freely selected. For example, classes that do not have a superordinate-subordinate relationship can be set in relation to each other using non-taxonomic relations. Attributes are assigned to individual classes. Such attributes can contain various data types, such as numerical values or texts.

Individuals (or, understood here synonymously, instances) are a concrete manifestation of a class. Individuals have exactly the attributes and relations of the class to which they belong; additional attributes and relations cannot be added at the individual level. Instead, concrete relation and attribute values are assigned to individuals at said level.

Both attributes and relations can be limited using restrictions. Restrictions are used to define semantic restrictions on classes and their properties. These restrictions are expressed using a logical descriptive language in order to restrict the assignment of attributes and relations to individuals and of individuals to classes. Further possibilities for describing relationships in an ontology using a logical descriptive language are, for example, set-theoretical links between classes, numerical quantifications, the hierarchization of relations, and the construction of inference and integrity rules. Inference rules are an essential component of knowledge-based systems and are used to infer new knowledge based on existing knowledge.

Figure 1 below illustrates the ontology components mentioned above using an example.

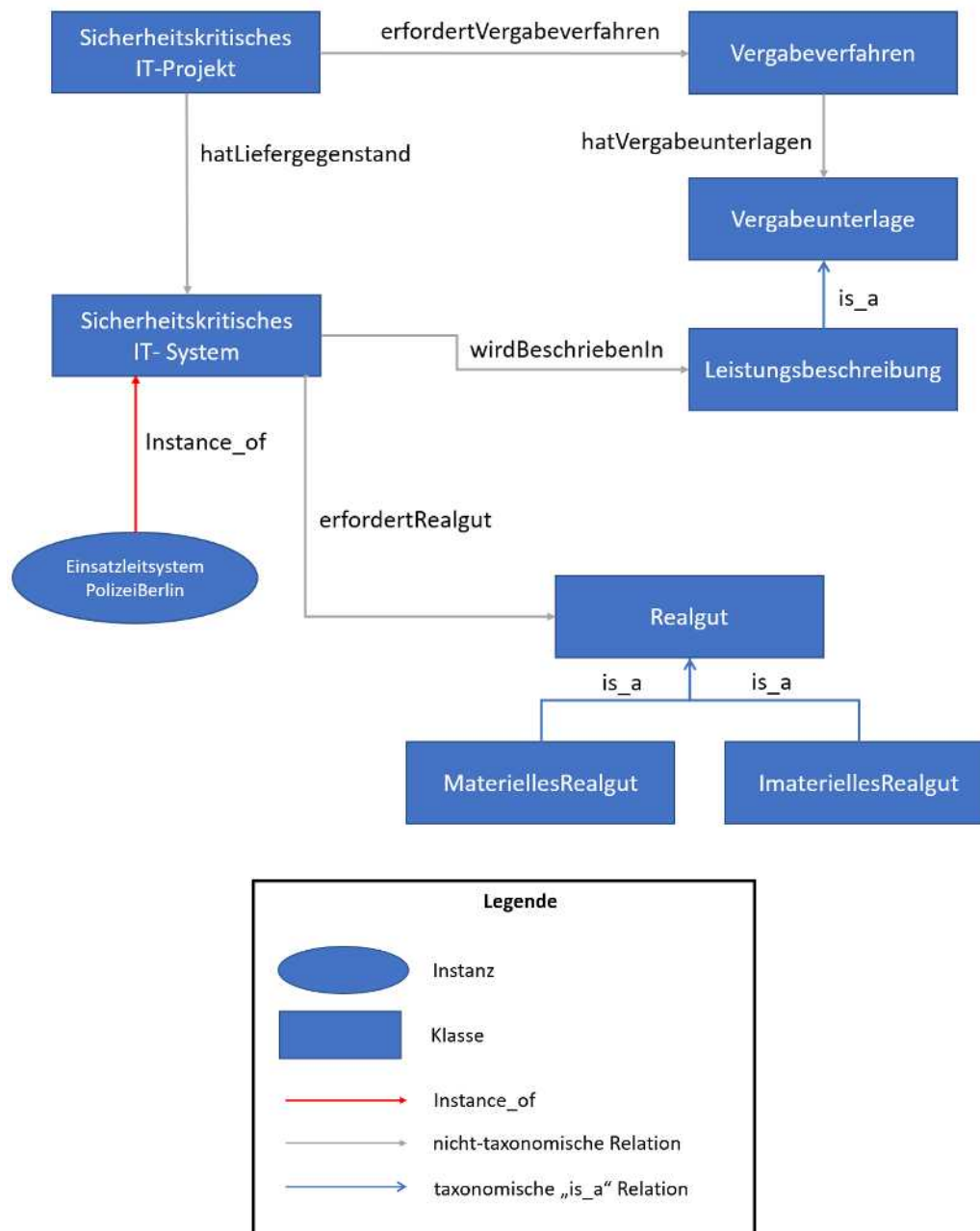


Figure 1: Example ontology to illustrate the ontology components

Figure 1 shows by way of example that the class *Leistungsbeschreibung* is a subclass of the class *Vergabeunterlage*. This means that all individuals that belong to the individual set of the class *Leistungsbeschreibung* are also summarized in the individual set of the class *Vergabeunterlage*. The inclusion context generally only applies in one direction. Therefore, in this context, the relation *is_a* is referred to as a subsumption relation. It is generally suitable for subordinating one class (*Vergabeunterlage*, *Realgut*) to another

class (*Leistungsbeschreibung*, *MateriellesRealgut* or *ImmateriellesRealgut*). The relationship between the classes *sicherheitskritisches IT-Projekt* and *Vergabeverfahren* is here made possible by means of the relation *erfordertVergabeverfahren*. The classes *sicherheitskritischesIT-Projekt* and *Vergabeverfahren* have attributes, namely *benötigt-Sicherheitsüberprüfung* and *unterliegtGeheimhaltung*, of the boolean data type.

A possible inference rule based on the example shown in Figure 1 is:

Inference rule in SWRL	Natural language translation
$\text{Vergabeverfahren}(\text{?vf}) \wedge \text{unterliegtGeheimhaltung}(\text{?ivf}, \text{?s}) \wedge \text{swrlb:contains}(\text{?s}, \text{True})$ $\rightarrow \text{sicherheitskritischesIT-Projekt}(\text{?its}) \wedge \text{benoetigtSicherheitsprüfung}(\text{?its}, \text{True})$	If a procurement procedure is subject to a confidentiality level, then a security-critical IT project requires a security check.

Table 1: Example of an inference rule

2.1.4 Ontology-related representation languages

Representation languages for ontologies enable a machine-readable representation of an ontology's components. As a result, they also provide scope for vocabulary control and the formal definition of terms. In the following, we present only the currently predominant Web Ontology Language (OWL) as a representation language. OWL is an ontology language introduced by the W3C, which is based on the Resource Description Framework (Schema) RDF and RDFS. OWL extends the syntax of RDF or RDFS with description logics, whereby RDF/RDFS is usually encoded in XML. Three variants can be distinguished in the OWL versions:

- OWL-Full enables the unrestricted use of all OWL language elements and thus contains OWL's full expressive power. The disadvantage of this high expressive power is that it leads to longer calculation times for conclusions.
- OWL-DL includes the complete OWL expressions. However, its restrictions vis-à-vis OWL-Full are aimed at reducing the calculation time of the conclusions.
- OWL Lite is a sublanguage of OWL-DL and primarily enables the expression of class hierarchies and simple restrictions. This sublanguage contains only the most elementary means of expression.

2.2 Case-based reasoning

2.2.1 Basic idea of case-based reasoning

Case-based reasoning (CBR) is a technique from artificial intelligence research that is used to “intelligently” solve general problems—or, synonymously, cases or projects—that can be traced back to the comparison between similar problems (“cases”). This technology provides design principles for the “intelligent” reuse of experience-based knowledge, which can be used to develop experience-based systems.

Case-based reasoning is based on two fundamental assumptions that are close to human problem-solving thinking. The first assumption is that similar problems have similar solutions. The second assumption is that although each problem is different, the type of problem is repetitive. The idea is that the problem solutions are stored in a database so that they can be used for the subsequent solution of new problems, if these are of a similar problem type.

From the perspective of business project management, a project—here synonymous with a problem or a case—always comprises three components: a project description, a project solution, and an evaluation of the project solution. A project’s specification is available in a project knowledge base, which is often also referred to as a case base, knowledge base, or project database.

The term “reasoning” relates to drawing conclusions about a new project to be completed (e.g., tendered) on the basis of old, already completed projects. In case-based reasoning, a new project—represented as a “case”—is compared with a collection of projects in the project knowledge base in order to find old projects (cases) that are as similar as possible. The most similar old project is used as a starting point for the development of an adaptation procedure for the current project. After processing, the current project and its project solution are transferred to the project knowledge base. The project knowledge base increases in size with each project processed. It therefore provides a “broad” basis in the search for a similar old project due to its constant growth. Case-based reasoning therefore “automatically” leads to the acquisition of experience-based knowledge about projects that have already been carried out. However, it may be that no project in the project knowledge base proves useful, i.e., has a required minimum similarity to the new project.

2.2.2 Case-based reasoning cycle

The CBR cycle, which is regarded as a typical process of case-based reasoning, can be traced back to AAMODT/PLAZA; cf. AAMODT/PLAZA (1994), pp. 44–45. AAMODT and PLAZA describe case-based reasoning as a cyclical process that comprises a total of four phases; cf. the following Figure 2.

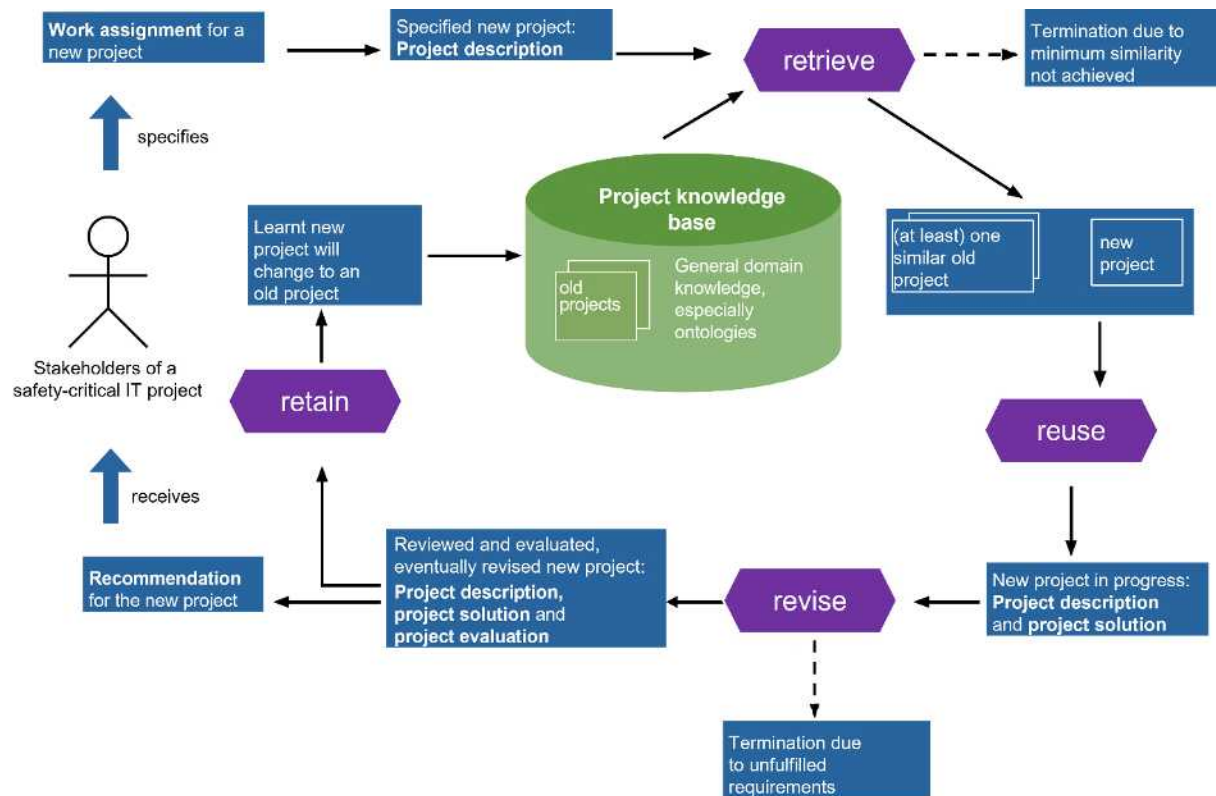


Figure 2: CBR cycle for knowledge reuse in operational project management

As shown in Figure 2, the CBR cycle begins with an action request for a new project. To solve the new project, those working on it must use the project description in the retrieve phase to search for at least one similar old, already processed project in the project knowledge base. If they find no old project in the retrieve phase that has a user-defined minimum similarity, they must abort the application of the CBR technique without a result.

In the reuse phase, those working on the project analyze the deviation in the project descriptions between the most similar old project identified and the requested new project. Based on the previously identified deviations, they adapt the solution of the similar old project to the new project's description. The result of this phase is a new project solution.

The subsequent review phase is used to check the new project solution and its suitability for reuse. For example, the suitability of the preliminary project solution can be assessed both manually, by experienced project managers, and with computer support, for example using integrity rules. If the project solution developed so far does not appear to be completely plausible, corrections can be made to the existing project solution. The aim of these corrections is to revise the preliminary project solution in such a way that all requirements for a plausible and reusable project solution are met. If the project solution cannot be fulfilled due to plausibility or reusability requirements, it is terminated due to “irreparable solution requirements”. Otherwise, the findings of the review phase are incorporated into the project evaluation.

Finally, in the retain phase, the triple from the project description, the project solution and the project evaluation for the new project are incorporated into the project knowledge base as knowledge. Parallel to the retain phase, the determined project solution and optionally also the project evaluation are issued to the project management as an action recommendation for the new project.

2.3 Project management of security-critical IT projects

2.3.1 Project management

Project management methods are generally used for the management of security-critical IT projects. Project management is defined as the entirety of management tasks, organizational forms, techniques, and resources for the initiation, definition, planning, control, and completion of projects. It describes the application of knowledge, skills, and techniques to project activities in order to fulfill project requirements. The project requirements should be met within the performance objectives of time, cost, quality, scope, benefit, and risk.

The frequently cited advantage of a project management standard is the use of a common language and thus a common understanding of project processing in order to reduce friction between the people involved in the project. In recent years, various project management standards have been established that are also suitable for the implementation of security-critical IT projects. The most common project management standards are the International Project Management Association (IPMA), Project Management Institute (PMI), Projects in Controlled Environments (PRINCE2), and Scrum.

The relevant specialist literature often distinguishes between traditional project management methods, such as IPMA, PMI, and PRINCE2, on the one hand, and agile project management methods, such as Scrum, on the other. However, it should be noted that

classic and agile project management methods do not have to be fundamentally contradictory—they can also complement each other. Hybrid approaches are therefore possible as well; cf. HILMER/KRIEG (2014), pp. 47–48; HABERMANN (2013), pp. 93–94. The selection of a project management method essentially depends on the project, the contractor’s way of working, and the client’s requirements. In the following explanations of PRINCE2 and Scrum, we emphasize the “reuse of experience” from the two project management methods. We base our focus on these two project management methods on the following considerations:

- PRINCE2 is the world’s most widely used project management method alongside IPMA and PMI; cf. ERNE (2019), p. 20. In PRINCE2, “knowledge” and “learn from experience” are important components.
- In the agile standard “Scrum,” the “Sprint Retrospective” is a central point for learning from experience.
- The agile approach is becoming increasingly important in the project management of safety-critical IT projects. One example is the company Eurofunk Kapbacher GmbH, which carries out safety-critical IT projects and describes the agile approach as the preferred standard when it comes to implementing deployment control systems. In some award procedures, clients explicitly request the use of the Scrum project management method to implement a safety-critical IT project. One example is the implementation of an operations control system for the police in the state of Brandenburg.
- The PRINCE2 and Scrum project management methods can be used together as a hybrid approach in a safety-critical IT project.
- A PRINCE2 and risk management ontology already exists that is designed for safety-critical IT projects and can be used as the basis for constructing an ontology for safety-critical IT projects. The PRINCE2 and risk management ontology is described in WEBER et al. (2021), pp. 12–23, and is reused here.

Based on the reasons described above, it seems sensible to combine the knowledge-based focus of PRINCE2 and Scrum with the practical approach of case-based reasoning for the reuse of experience-based knowledge from safety-critical IT projects.

2.3.2 Project management domain: Security-critical IT projects

No standardized definition of security-critical IT projects currently exists. In particular, it should be noted that the definition of a security-critical IT project can vary depending on the industry and company environment. However, regardless of the industry, it is necessary to carry out a risk analysis to ensure that the right measures are taken to protect the IT systems and data an IT project is to develop and provide. In this article, we focus on *public projects* that are financed by public funds and awarded through a public contract.

In order to define a safety-critical IT project, we must first define the terms “project”, “IT project”, and “safety-critical”.

PRINCE2 defines a project as “a temporary organization that is created for the purpose of delivering one or more business products according to an agreed business case” (AXELOS (2015, p. 33). A project therefore has the following characteristics (cf. AXELOS (2018), pp. 8–9):

- Change: A project should realize a change.
- Temporary: A project is a temporary undertaking.
- Cross-departmental: Work on a project is carried out by a team of people with different skills.
- Unique: The project’s intention is unique.
- Uncertain: the project brings opportunities, but also threats in the form of risks.

An IT project has the same characteristics as the project term explained above. However, an IT project deals specifically with the implementation of an information technology system.

Safety-critical IT projects have all the aforementioned characteristics of an IT project and also deal with the implementation of a safety-critical IT system. Safety-critical IT systems (also known as mission-critical systems) are found in critical infrastructure organizations and require “functional safety”; cf. the international standard IEC 61508. The federal government defines critical infrastructure organizations as “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 INNEREN (2009), p. 3). In Germany, organizations and facilities in the areas of energy supply, information technology and telecommunications,

transport and traffic, health, water, food, finance and insurance, government and administration, as well as media and culture are counted as critical infrastructures.

A project failure or an IT malfunction in the subsequent operation of a safety-critical IT system, which indicates a lack of work in the safety-critical IT project, would have a far-reaching impact. These impending consequences make the early identification and assessment of risks a vital task within a safety-critical IT project. Risks are identified and assessed continuously and independently of the respective project phase. Due to the risks and their imminent consequences, safety-critical IT projects must be managed differently to normal IT projects; cf. GASSMANN (2001), p. 12.

The concept of risk occupies a central position within a safety-critical IT project and requires a special form of risk management. The risk management of a safety-critical IT project must identify potential risks as “weak signals” at an early stage and correctly interpret them in order to assess the impending effects and plan any necessary counter-measures in advance.

In summary, this article defines a safety-critical IT project as an IT project that has the following specific characteristics:

- A safety-critical IT project is a project in which the integrity, confidentiality, and availability of the safety-critical IT system to be implemented are of central importance and whose protection requirement category is therefore classified as “very high”.
- A safety-critical IT project requires functional safety to ensure a high level of protection against imminent damage caused by malfunctions or undesired behavior of the safety-critical IT system.
- Due to the identification as critical infrastructure and the associated commercial dependency on the IT system, legal requirements apply.
- One of the central stakeholders of the project is a KRITIS organization.
- The project is subject to an above-average level of risk.
- A failure or loss of data in the safety-critical IT system to be implemented can have a serious impact on social order.
- A contract for a security-critical IT project is awarded to a bidder through an upstream public procurement procedure.

Due to the aforementioned characteristics, a special form of risk management and the associated early identification of risks in the form of an interpretation of “weak signals” are necessary.

2.4 IT applications

2.4.1 Monolithic applications

The term “monolithic application” is not uniformly defined in information technology. However, a “monolith” is understood to be an IT application (software) whose functional elements are not separated from each other and are provided as a code block. The software application is self-contained, independent of other applications, and usually classified in two- or three-tier architectures. The three-tier architecture distinguishes between the data storage level, the technical concept level, and the presentation level. In a two-tier architecture, the business concept level is omitted.

A monolithic application is characterized by the following features:

- All functionalities and components of a monolithic application are traditionally implemented in a single code base.
- A monolithic application generally uses a database to store and retrieve data.
- Further development of the application function in a monolithic application is usually carried out step by step.
- Monolithic applications can be more difficult to scale in an operational environment because they are built on a code base and cannot be scaled horizontally.
- In practice, monolithic applications often require a longer development time, as all application functionalities are implemented in a single code base and must be tested as a complete application.

2.4.2 Cloud-native applications

The term “cloud-native application” (also: “cloud-native computing”) is not uniformly defined. Particularly noteworthy is the definition by KRATZKE (2022), p. 35, which says a cloud-native application is: “a distributed, observable, elastic service-of-services system optimized for horizontal scalability that isolates its state in (a minimum of) stateful components.” In simplified terms, this means that the software application is designed for the cloud according to design-oriented principles and is then deployed and executed in a cloud environment. The following basic characteristics are attributed to a cloud-native application; cf. VETTOR/SMITH (2023), pp. 5–10; CALDATO (2020), pp. 1–2:

- The software application is developed exclusively in a cloud and is designed for exclusive use in said cloud.
- Cloud-native applications use open source technologies and are primarily geared towards transparency and interoperability.
- The cloud-native approach focuses on creating functionalities that are based on serverlessness and can be deployed as encapsulated microservices.
- Cloud-native applications are designed for horizontal scaling.

Current studies consider both cloud-native applications and the development of these applications, which also takes place in a cloud, to be state of the art; cf. DELOITTE (2022), pp. 28–34; SLASHDATA (2021), pp. 13–16; CAPGEMINI (2021), p. 36; GARTNER (2021), p. 7; LÜNENDONK (2021), pp. 7–8 and 13–36, in particular pp. 17–36. This development is largely driven by the development of cloud technologies.

This article provides an exemplary “click prototype” for the cloud-native application and implements various serverless functions. The main function processes the result and returns it as a response to the request to the API gateway, which in turn forwards it to the requested client. Architecturally, the example shown above can be represented in an Amazon Web Service (AWS) cloud as follows.

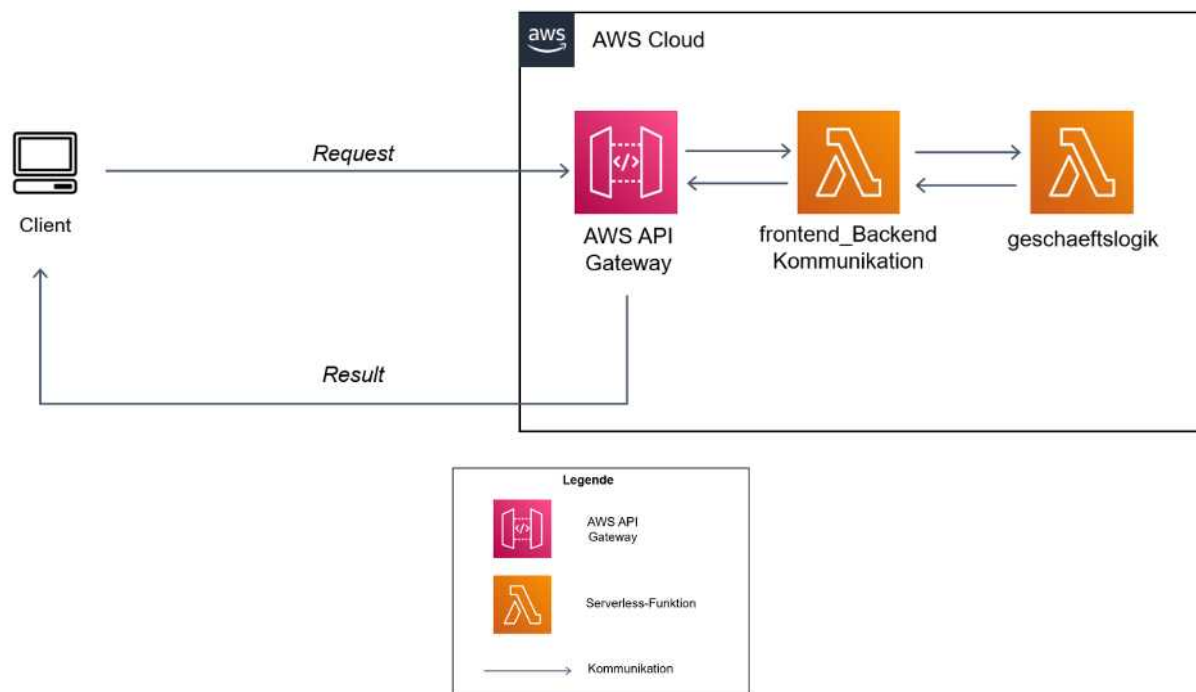


Figure 3: Calling a serverless function in the Amazon Web Service (AWS) Cloud

3 Application of ontology-supported case-based reasoning for the reuse of empirical knowledge about safety-critical IT projects

3.1 Procedure for the application of ontology-supported case-based reasoning for the reuse of empirical knowledge about safety-critical IT projects

For the application of ontology-based case-based reasoning for safety-critical IT projects, five steps are helpful. They build on each other and are as follows (the terms “project” and “case” are used synonymously in this article):

1. Construction of a safety-critical IT project ontology
2. Integration of the safety-critical IT project ontology into the CBR tool jCORA
3. Description and modeling of projects to represent practical examples
4. Similarity calculation between projects using the ontology-based CBR tool jCORA
5. Adaptation of the solution of (at least) one similar old project to a new project as a follow-up problem—this is only outlined in this article but not dealt with in depth, but requires further research work.

We describe these steps in more detail in the following chapters, and they serve as a procedure for the application of ontology-supported case-based reasoning for safety-critical IT projects.

3.2 Construction of a security-critical IT project ontology

3.2.1 Selection of a design method for the development of a safety-critical IT project ontology

The specialist literature lists various construction methods for creating an ontology; cf. GÓMEZ-PÉREZ/FERNÁNDEZ-LÓPEZ/CORCHO (2004), pp. 113–154. However, there is no standard procedure for creating an ontology.

In this paper, we base our ontology construction on the method of NOY/MCGUINNESS; cf. NOY/MCGUINNESS (2001), pp. 4–23. The authors propose seven consecutive activities for the construction, which we here extend with a further activity—the definition of rules (Semantic Web Rules). This extension is based on the fact that, in addition to cardinalities, semantic web rules are also defined for the safety-critical IT project ontology, which enable further reasoning. However, NOY/MCGUINNESS do not provide for such a rule development in their design method.

The eight activities for the construction of an ontology relevant to this article are as follows:

1. Determine the domain and scope of the ontology
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define the properties of classes
6. Declare the properties by means of cardinalities
7. Define Semantic Web Rules
8. Create global individuals

When implementing the above procedure in practice, repeating an activity may prove necessary. The “linear”, consecutive implementation of these activities is sometimes difficult, as previous activities must often be adapted during the ontology’s construction. Therefore, the eight activities mentioned above should rather be understood as an idealized procedure for ontology construction and can include further sub-activities within an activity in addition to running through an activity several times.

Furthermore, it is important to understand that the ontology to be created is a basic product in the form of an OWL file and is not an independent software application. An ontology can be expanded or modified at a later date. This may be necessary if the user perspective changes or if more far-reaching aspects, such as “radically” new projects, need to be taken into account.

3.2.2 Selecting Protégé as an ontology editor

Various ontology editors are available for constructing an ontology. The ontology editors frequently mentioned in the literature are, for example, Apollo, OntoStudio, Swoop, and Protégé. The use of an ontology editor is not mandatory. However, it is generally recommended for the construction of extensive ontologies. Ontology editors offer tool-supported construction aids and enable the validation of the constructed ontology by means of a reasoner. Furthermore, there is extensive documented help to facilitate the use of an ontology editor.

We have selected the ontology editor Protégé version 5.5.0 for the development of the safety-critical IT project ontology. We have based this selection on the following arguments:

- Several studies, such as BEIBEL (2011), pp. 85–112, compare different ontology editors, with the ontology editor Protégé proving a recommendable alternative on several occasions. BEIBEL's very detailed study assessed ontology editors in terms of their functionality, reliability, and usability.
- There are various sources and online documentation in the form of video recordings to help potential users familiarize themselves with the software. Online help is also available for the Protégé ontology editor.
- Various modules can be installed as “plug-ins”. The expandability ranges from additional visualization options to additional reasoning components.

However, there are also disadvantages that we have considered initially acceptable for this article:

- The ontology editor contains errors that sometimes cause the program to crash.
- Protégé's usability is considered in need of improvement, as it is difficult to intuitively apply the tool without first reviewing documents.
- It is a desktop-based system requiring a Java runtime environment.

Protégé is an ontology editor developed at Stanford University. It is currently subject to the General Public License and is freely available, including the source code. Protégé supports two types of ontology creation, namely a frame-based and an OWL-based approach. The frame-based approach makes possible the creation of classic ontology components, such as classes and relations. The OWL-based approach, which is used in this article, makes it possible to fully utilize the expressive power of OWL and RDF/RDFS.

Protégé offers the possibility to extend its functionality with plug-ins. The following plug-ins are relevant for this article:

- OntoGraf
- OWLViz
- SWRL-Tab

The two plug-ins “OntoGraf” and “OWLViz” enable additional visualizations of an ontology. The “SWRL-Tab” plug-in is used to support the construction of rules.

In addition to the already mentioned components, the integrated inference mechanisms contained in Protégé, the so-called reasoners, deserve special mention. Examples include Fact++, Pellet, and HermiT. These reasoners can be used to derive implicit knowledge. In addition, reasoners make it possible to check an ontology's logical consistency.

3.2.3 Application of the construction method for the construction of a safety-critical IT project ontology

3.2.3.1 Defining the scope of an IT project ontology

We here define the section of the application area (domain) of an IT project ontology that is to be covered. We set out to determine which aspects of said application area should be described in the ontology and which areas should be neglected. In addition, these steps should define the ontology's level of detail and purpose.

In order to define the application's scope, we first formulate basic questions. Table 2 below shows these questions for the safety-critical IT project ontology to be created.

Basic questions	Answers
Which domain does the ontology refer to?	The ontology refers to the application domain of safety-critical IT projects.
For what purpose is the ontology is used?	The ontology's purpose relates to the structuring and representation of domain-specific knowledge for safety-critical IT projects. It serves as a specification of commonly used linguistic expressions to improve communication between actors collaborating in safety-critical IT projects. It functions as the basis for the prototype CBR tool jCORA and for use in a cloud-based, ontology-supported CBR system to support the reuse of empirical knowledge from safety-critical IT projects.
Who should use the ontology?	The safety-critical IT project ontology is intended to be used by all stakeholders involved in safety-critical IT projects. This includes, in particular, public authorities and organizations with security tasks that award security-critical IT projects, as well as contractors that carry out such projects.

Table 2: Basic questions to narrow down the application's scope

Although the basic questions provide a rough framework, they alone are not sufficient to specify the application's scope. The formulation of competency questions is often suggested for concretization. Formulating these competency questions is aimed at allowing the ontology designer to focus on answering them when specifying the ontology. From this designer's perspective, answering the competency questions represents the areas of particular interest. In addition, these questions should be understood as expected requests from the ontology's potential users. One should be able to fully formulate the competency questions using the linguistic means of expression provided in the safety-

critical IT project ontology. One should also be able to develop a question list that is as complete as possible, as this makes it possible to check that the relevant aspects are covered and that the ontology is correct and complete.

In order to formulate practical competency questions, we interviewed six experts who work on safety-critical IT projects in various roles in this field. These experts are project managers who, according to their statements, each have more than five years of project experience with a total project volume of more than 5 million euros in the integration of safety-critical IT systems. We will mention the following competence issues as results:

- What types of risk exist in safety-critical IT projects?
- Who are the stakeholders of a safety-critical IT project?
- What characteristics define the availability of a safety-critical IT system?
- How is the fault tolerance of a safety-critical IT system measured?
- What does the structure (system environments) of a safety-critical IT system look like?
- What influence do changes in legislation have on safety-critical IT projects?
- What skills are required for the implementation of safety-critical IT projects?
- What does a specification sheet (defined scope of delivery) look like for safety-critical IT projects?
- What is the demarcation/embedding/interaction with other safety-critical IT projects and environments?
- What does the necessary increased quality assurance for security-critical IT projects look like?
- Which methods can be used for risk assessment?
- What security levels exist for security-critical IT projects?
- Which KRITIS sectors can have security-critical IT systems?
- Which availability classes exist for safety-critical IT systems?
- What risk measures exist for safety-critical IT projects?
- How is the forward-looking consideration of the state of the art (current, future) carried out, especially for long-term IT projects?
- What protective measures are in place for security-critical IT systems?
- What influencing factors exist in security-critical IT projects?

- What does a project organization look like for security-critical IT projects?
- What is the difference between a “normal” IT project and a safety-critical IT project?
- Which documents are necessary for the traceability of a safety-critical IT system?
- Which components are relevant for the traceability of a safety-critical IT project and safety-critical IT system?
- Where and with which characteristics is the usability of a safety-critical IT system measured?
- What are the characteristics of a safety-critical IT system?
- What implementation strategies exist in safety-critical IT projects?
- What impact does the tendering process have on the subsequent realization of a project?
- What does the calculation of a safety-critical IT project look like?
- Which components are relevant for the calculation?
- Which documents are relevant for a safety-critical IT project?
- How are the cost and time frame defined in a safety-critical IT project?
- Which safety mechanisms can be used in a safety-critical IT system?
- Which risks need to be tracked to ensure the project’s success?
- How can risks be assessed and managed?
- What restrictions must be taken into account in the tendering process?
- How is the tendering law complied with?
- What changes are required to the original project scope, and how are these integrated into said scope?
- Which project elements can be firmly priced without taking on too much risk?
- What are the success factors in security-critical IT projects?
- What are the failure factors in security-critical IT projects?
- What risks exist in safety-critical IT projects?
- What are the components of a safety-critical IT system?

- Which political stakeholders must be taken into account in order not to jeopardize the project's success?
- What project justification exists in the public interest, e.g., ensuring the supply of services to citizens?
- How can a safety-critical IT project be carried out with the help of PRINCE2?
- What plans need to be drawn up for the implementation of a safety-critical IT project?
- Which requirements (functional, non-functional, and technical) must be taken into account in order to build a safety-critical IT system?
- Are there requirements for the introduction and operation of a safety-critical IT system?
- What should a concept or IT architecture for a safety-critical IT system look like?
- How can modern and future-oriented cloud technologies be used in a safety-critical environment while continuing to meet security requirements?
- How and when do employees need to be trained to adequately ensure awareness of the risks?
- How can user acceptance of a new IT system be ensured?
- What data is required in a security-critical IT system?
- How can the availability and integrity of the data be ensured?
- How can the high availability of a safety-critical IT system be ensured?
- How can it be ensured that a safety-critical IT system is efficient and appropriate despite redundancy requirements?
- What documentation is expedient to ensure the operational readiness of a safety-critical IT system?
- How can the operational readiness of a security-critical IT system be ensured with the help of a contractual agreement (according to EVB-IT)?
- What fault tolerance must a security-critical IT system have?

The list of competency questions presented must always be taken into account during the development process of the safety-critical IT project ontology and expanded or shortened if necessary. The subsequent removal of a competency question is possible if

it subsequently proves irrelevant. In this article, however, we have removed no competency questions; rather, further competency questions arose over the course of our ontology construction. For example, further competency questions proved necessary with regard to the award procedure, which impacts the implementation of a safety-critical IT project. Competence issues such as “What restrictions must be taken into account in the tendering process?” and “How is the tendering law complied with?” do exist. However, a closer look at the aforementioned competency issues leads to further questions, such as “Which procurement law can be applied to security-critical IT projects?”. This example illustrates the need to formulate further questions as part of the ontology construction.

3.2.3.2 Testing existing ontologies for their use in the construction of a new IT project ontology

In this step, we consider whether we can reuse a similar existing ontology as a foundation, either in full or in part.

Various “ontology libraries” are available for searching for potentially reusable ontologies. Examples include DAML and the Protégé Ontology Library. We found no ontology in these libraries that was reusable for the development of an ontology for safety-critical IT projects. However, two other ontologies—ones outside of the relevant ontology libraries—did prove suitable for reuse in this article.

The development of a safety-critical IT project ontology is based on a project management domain ontology (PM domain ontology for short), which was developed by the Institute for Production and Industrial Information Management (PIM) at the University of Duisburg-Essen as part of the BMBF-funded joint project KI-LiveS (AI Laboratory for Distributed and Embedded Systems). We will integrate the safety-critical IT project ontology to be created into the PM domain ontology as a sub-area.

Using the PM domain ontology as the basis for the construction of the safety-critical IT project ontology has the following advantages for this paper:

- Basic classes, relations, and attributes of project management already exist in the PM domain ontology and are defined in very general terms, allowing us to make a concrete classification of terms as subcategories in the sense of a taxonomy.
- The PM domain ontology has already been tested using the CBR tool jCORA.
- User-specific customization is possible without prior adaptation measures.

- As part of the KI-LiveS project, the PM domain ontology was used to develop a PRINCE2 and risk management ontology with a focus on safety-critical IT projects.

Figure 4 below shows the scope of the PM domain ontology in Protégé used for this article.

Ontology metrics:	
Metrics	
Axiom	1576
Logical axiom count	999
Declaration axioms count	577
Class count	357
Object property count	180
Data property count	37
Individual count	0
Annotation Property count	0

Figure 4: PM domain ontology

However, we have also identified the following disadvantages of using the PM domain ontology:

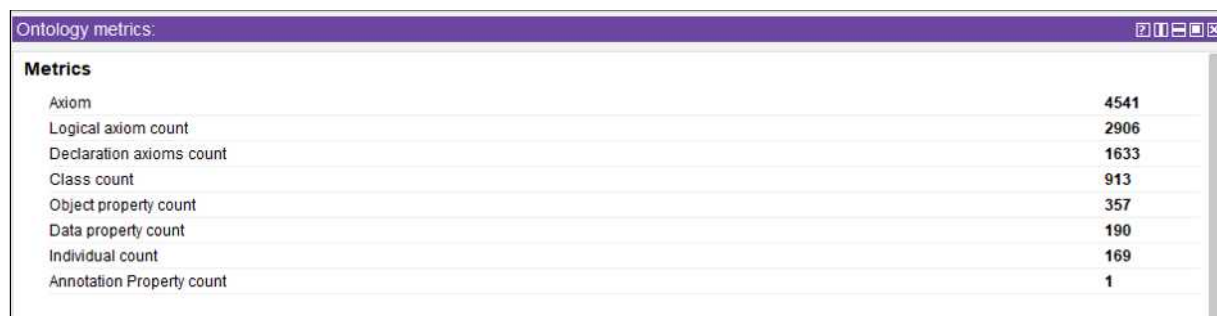
- No meaningful documentation for the PM domain ontology's components currently exists.
- The PM domain ontology is a very extensive ontology, comparable to an "upper ontology". The ontology's complexity requires a non-negligible training period, which, as already mentioned, currently has to take place without meaningful documentation.
- Cardinalities and semantic rules are not defined in the PM domain ontology. Therefore, one point of criticism is that the PM domain ontology has not yet been sufficiently specified.
- The PM domain ontology is subject to continuous development, which means that classes, relations, and attributes can change at any time. There is a lack of version information on the ontology, making versioning difficult.

Overall, however, the advantages mentioned above outweigh the disadvantages mentioned here. The disadvantages are deliberately accepted in order to extend the PM domain ontology and the associated OWL file with the ontology components of the safety-critical IT projects in Protégé.

Alongside the PM domain ontology, we also use the PRINCE2 and risk management ontology, which was also developed as part of the KI-LiveS project. This ontology was created on the basis of the PM domain ontology and is specifically designed for safety-critical IT projects. Reusing the PRINCE2 and risk management ontology to construct an ontology for safety-critical IT projects has the following advantages:

- The ontology has been developed for safety-critical IT projects.
- The ontology is based on the PM domain ontology.
- The ontology was used to create example cases for safety-critical IT projects in the CBR tool jCORA.
- The use of global individuals saves time when modeling IT projects.
- The PRINCE2 and risk management ontology is documented in detail in the publication WEBER et al. (2021), pp. 15–23 and 50–75.

Figure 5 below shows the scope of the PRINCE2 and risk management ontology for safety-critical IT projects used for this article.



Ontology metrics:	
Metrics	
Axiom	4541
Logical axiom count	2906
Declaration axioms count	1633
Class count	913
Object property count	357
Data property count	190
Individual count	169
Annotation Property count	1

Figure 5: Scope of the PRINCE2 and risk management ontology

However, the PRINCE2 and risk management ontology is subject to the following restrictions, which must be taken into account when constructing a safety-critical IT project ontology:

- Not all of the ontology's terms originate directly from the PRINCE2 standard.
- Not all of the PRINCE2 standard's areas and terms are sufficiently taken into account in the ontology.
- Comments by technical experts, for example in the form of defined competency questions or terms, are missing.
- Integrating the PRINCE2 and risk management ontology into the underlying PM domain ontology can lead to contradictions, because when constructing the

PRINCE2 and risk management ontology, some classes were removed from the underlying PM domain ontology that are important for expressiveness in relation to safety-critical IT projects.

The objective in constructing the PRINCE2 and risk management ontology was to design the ontology for safety-critical IT projects. Due to this background, various basics such as the integration options in the CBR tool jCO-RA using case studies from the safety-critical IT project environment and the consideration of some (if not all) safety-critical IT project features have already been taken into account in this PRINCE2 and risk management ontology. Although some limitations in the ontology construction exist, as the listed disadvantages show, we have corrected them in this article.

3.2.3.3 Definition of important terms for the IT project ontology

We are developing a list of terms that is as complete as possible and relevant for the safety-critical IT project ontology. To practically construct an ontology for safety-critical IT projects, we asked experts about relevant terms from the domain of safety-critical IT projects, and extracted terms from relevant service descriptions of public tenders that are aimed at safety-critical IT projects.

When listing the relevant terms, it is irrelevant from the perspective of an ontology whether they are potential classes, attributes, or relations. The collection serves as a list of relevant terms that are to be used in the construction process of the ontology. In this step, the list does not yet contain any specifics for the terms' later use. Rather, the aim is to obtain a comprehensive list of what needs to be taken into account in the ontology to be created. Table 3 below shows an example of the terms identified for safety-critical IT projects.

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	Misserfolgskfaktoren	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

Table 3: Terms relating to safety-critical IT projects (exemplary excerpt)

Although we are not providing a complete list of terms from the PRINCE2 standard, we do give an exemplary list of risk management terms that played a major role in the aforementioned expert interviews below.

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		

Table 4: Risk management terms (excerpt as an example)

The terms shown in Table 3 and 4 are essential linguistic means of expression that were considered necessary both by the experts interviewed and in the relevant service descriptions in order to conceptualize the application area “safety-critical IT project” in linguistic terms.

3.2.3.4 Class construction

For a better understanding of the classes of the safety-critical IT project ontology we later present as examples, we must first make some preliminary remarks on our approach to class construction, for the following points:

- Explanation of the class construction
- Relevance of the terms for class construction
- Selection of the method for class construction
- Chapter structure for the explanation of the constructed classes
- Table structure for the explanation of the constructed classes

As far as possible, we base our explanation of the class construction on the classes constructed in this article. We do not provide an explanation of the class from the underlying PM domain ontology and the PRINCE2 and risk management ontology for all classes; we explain only those that we additionally constructed as part of this article. However, in individual cases, classes from the PM domain ontology must be explained if an extension is made in the taxonomic sense in order to justify classification in the class. The explanation includes how the content of the existing class from the PM domain ontology was designed in order to justify the design decision. In principle, we did not change the first and second hierarchy levels of the PM domain ontology because we deemed no changes as necessary for these hierarchy levels.

The collection of terms mentioned in chapter 3.2.3.3 serves as the basis for class construction. For this purpose, the terms from the term collection that are to be represented by classes in the ontology must be identified. Terms with a high level of detail should be considered as individuals. More general terms should be defined as classes. However, practical ontology construction shows that the distinction made is ultimately subjective and can therefore be justified in different ways depending on the decisions of the ontology constructor. The decision as to whether a term is modeled as a class or as an individual is made on a case-by-case basis.

Although the terms from chapter 3.2.3.3 are used as a basis for the construction of the classes, we also construct classes in this article that have not previously been mentioned

as terms. The terms serve as an orientation and construction aid, but do not represent the exclusive means for class construction. As already described, the performance descriptions taken as a basis, in addition to the terms mentioned by the experts, represent an essential contribution to the class construction.

The literature contains various methods for said class construction. The following three merit frequent mention (cf. NOY/MCGUINNESS (2001), pp. 6–7):

- Top-down approach: The most general domain classes are defined first. The other subclasses are then constructed until the class hierarchy's lowest level is reached.
- Bottom-up approach: The first step is to define the most concrete class. The more concrete classes on the lower hierarchy levels are grouped together until the most general class is reached.
- Middle-out approach: The top-down and bottom-up approaches are combined.

A combination of the top-down and bottom-up approaches is used to construct the safety-critical IT project ontology. This is justified by the fact that the underlying PM domain ontology and the PRINCE2 and risk management ontology already specify a class structure and a combined approach is considered more expedient. An exclusive top-down or bottom-up construction was not always possible in the practical construction of the safety-critical IT project ontology. In the course of ontology construction, further findings emerged—e.g., during subsequent case creation, which made it necessary to adapt the safety-critical IT project ontology. In practical implementation, it has proven practicable to first define the most important classes.

However, for a comprehensible ex-post explanation of the safety-critical IT project ontology, it seems appropriate to choose the top-down approach to make it easier to understand the class constructions, which we explain in tabular form. The respective tables are structured as follows:

Class	Description	Subclass of

Table 5: Table structure for the description of classes

The constructed or reused class is described in the “Description” column and the parent class is named in the “Subclass of” column.

In the following, we describe only a selection which, in our opinion, provide an easily understandable overview of the ontology's class structure for safety-critical IT projects presented here, especially for readers without extensive experience with ontologies. Readers interested in further details of this ontology are referred to the comprehensive documentation of the class structure in SETHUPATHY (2024), pp. 102–234 and 645–677.

The class *Thing* is the maximum class or “top” class of the safety-critical IT project ontology. All subsequent classes are subclasses of the maximum class *Thing*. The class *Thing* represents the taxonomy's starting point and forms the “zeroth” hierarchy level. The class *Thing* determines that the ontology contains only a single taxonomy. The existence of multiple, parallel taxonomies is excluded by the use of the class *Thing*.

The first subclasses of the class *Thing* are the classes *Eigenschaft* and *Objekt*.² Both subclasses originate from the PM domain ontology. Figure 6 below shows the maximum class *Thing* and the directly subordinate subclasses *Eigenschaft* and *Objekt* on the first hierarchy level.

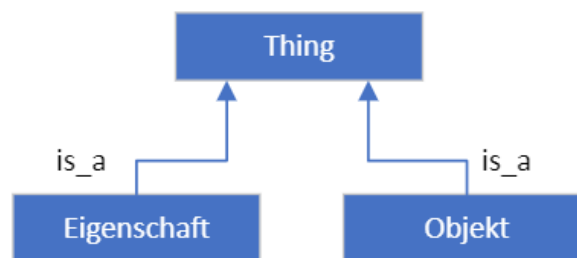


Figure 6: Subclasses of the class *Thing*

The class *Eigenschaft* is first explained using selected subclasses as examples. Later, the class *Objekt* is described with exemplary selected, associated subclasses.

The class *Eigenschaft* is specified in the first hierarchy level by the utilized PM domain ontology and is characterized in the safety-critical IT project ontology as follows:

2) In this English-language article, all designations of linguistic expressions for the ontology-supported CBR system, which was created using the CBR tool jCORa, are reproduced in German as they were implemented in the underlying ontologies (the PM domain ontology, the PRINCE2 and risk management ontology and the security-critical IT project ontology based on these) and are used in the ontology-supported CBR system presented here. The German-language terms in the original are deliberately retained because numerous linguistic, in particular semantic, subtleties can only be reproduced unadulterated in the (German) language originally used. In addition, these German-language terms, which have been retained in the original, are generally formatted in italics to emphasize their “linguistically frozen” character, among other things.

- The class *Eigenschaft* must be distinguished from the ontology components of attributes and relations, which are also referred to as a class's characteristics. The general construction of classes for the safety-critical IT project ontology shows that it is not trivial to differentiate between subclasses of the Property class and the characteristics (in the form of attributes and relations) of a class. In individual cases, the distinction can be fluid and is subject to the ontology constructor's subjectivity. This fluid distinction is exemplified by the subclasses of the class *InformationstechnischeEigenschaft*. These were designed as sub-subclasses of the class *Eigenschaft* but could also have been constructed as properties by using primitive data types within a class, such as the class *Hardware*. This example is intended to illustrate the challenge of differentiation at this point and emphasizes an ontology's subjective nature and its construction ("design") as a creative process that cannot be fully "objectified".
- The subclasses of the class *Eigenschaft* provide linguistic means of expression for the property descriptions for the subclasses of the class *Objekt*, which are linked by means of non-taxonomic relations and are of particular importance for the safety-critical IT project ontology. Properties worth mentioning are, e.g., information technology properties, environmental conditions, and award procedure types. These properties are used to model special characteristics that cannot be adequately expressed with an attribute.
- Properties are characterized by their immutability, whereas objects can change.

Subclasses of the class *Eigenschaft* are discussed below.

The class *Eigenschaft* is the PM domain ontology divided into the three subclasses *InstanzBeschreibungskonstituente*, *QualitativeEigenschaft*, and *QuantitativeEigenschaft*, as shown in Figure 7 below. The subclasses represent the second hierarchical level of the safety-critical IT project ontology.

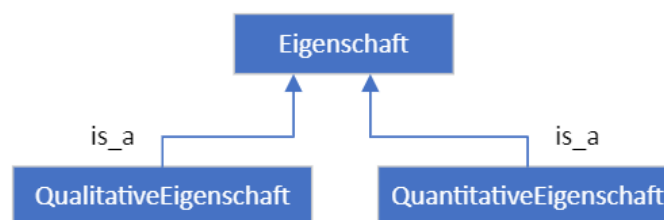


Figure 7: Second hierarchy level of the class *Eigenschaft*

The distinguishing criterion between the three subclasses is whether the property is based on a natural language feature component or whether said component consists of a measurable key feature. The class *Eigenschaft* and the associated subclasses are explained in Table 6 below.

Class	Description	Subclass of
QualitativeEigenschaft	The class serves as a superclass for all qualitative properties. Qualitative properties are all non-numerical, naturally linguistic properties.	Eigenschaft
QuantitativeEigenschaft	The class serves as a superclass for all quantitative properties. Quantitative properties are all numerical properties that can be measured by means of a key figure.	Eigenschaft

Table 6: Description of the subclasses of the class *Eigenschaft*

The class *QualitativeEigenschaft* contains various subclasses, as shown in Figure 8 below.

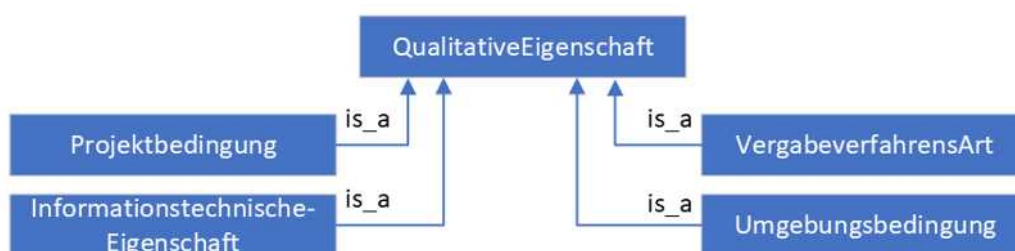


Figure 8: Subclasses of the class *QualitativeEigenschaft*

In this third hierarchy level, the classes *InformationstechnischeEigenschaft*, *Umgebungsbedingung*, *Projektbedingung*, and *VerfahrensArt* are added as subclasses of the class *QualitativeEigenschaft*. The classes mentioned are described in more detail in Table 7 below. They are mainly used to structure generally valid qualitative characteristics of project conditions, public award types, and environmental conditions with a focus on safety-critical IT projects.

Class	Description	Subclass of
Projektbedingung	The class represents class represents a prerequisite for a project that serves as an internal company condition and, in contrast to the environmental condition, does not reflect any influence from outside the company.	QualitativeEigenschaft
Informationstechnische Eigenschaft	The class subsumes various information technology properties that are relevant for safety-critical IT projects.	QualitativeEigenschaft
Umgebungsbedingung	The class subsumes various environmental conditions that can affect a safe-ty-critical IT project as external factors. These environmental conditions serve to take into account a project's "soft" factors. In this context, an environmental condition represents a state of external influences.	QualitativeEigenschaft
VergabeverfahrensArt	There are various types of public procurement in Germany. The current types of award procedures are described using this class.	QualitativeEigenschaft

Table 7: Description of the subclasses of the class *QualitativeEigenschaft*

The class *QuantitativeEigenschaft* contains the classes *DerivateQuantitativeEigenschaft* and *OriginäreQuantitativeEigenschaft*. Both classes are already defined by the PM domain ontology. Figure 9 shows the class *QuantitativeEigenschaft* with the named subclasses.

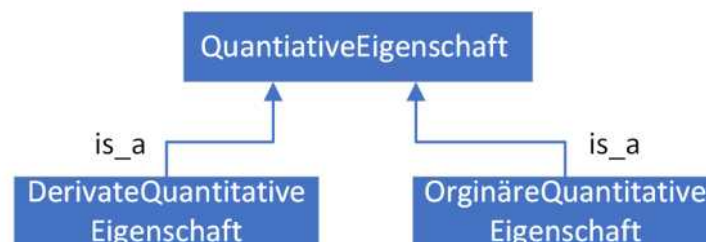


Figure 9: Subclasses of the class *QuantitativeEigenschaft*

The two subclasses of the class *QuantitativeEigenschaft* are used in the safety-critical IT project ontology to distinguish between derived metrics and original metrics. As safety-critical IT projects are awarded to a bidder through a public tender, the derived key figures—such as those from suitability criteria, exclusion criteria, and award criteria—should also be reflected in the class construct. Quantity structures as derived key figures also play an important role for security-critical IT projects. Original quantitative properties, on the other hand, are simple key figures. For example, quantities are used for the dimensioning of IT systems. The key figures are regarded as factual and differ from derived quantitative properties in that they are not used for a derived evaluation.

Class	Description	Subclass of
Derivative Quantitative Eigenschaft	The class <i>DerivativeQuantitative Eigenschaft</i> subsumes measurable properties that are represented by means of key figures, whereby the key figures' primary purpose is to derive an evaluation (e.g., suitability criteria, exclusion criteria, award criteria).	QuantitativeEigenschaft
Originäre Quantitative Eigenschaft	The class <i>OriginäreQuantitative Eigenschaft</i> comprises measurable characteristics that are presented using key figures without the intention of deriving a valuation from them.	QuantitativeEigenschaft

Table 8: Description of the subclasses of the class *QuantitativeEigenschaft*

The class *InformationstechnischeEigenschaft* describes the relevant properties of a security-critical IT system. Figure 10 below shows this class with the associated subclasses.

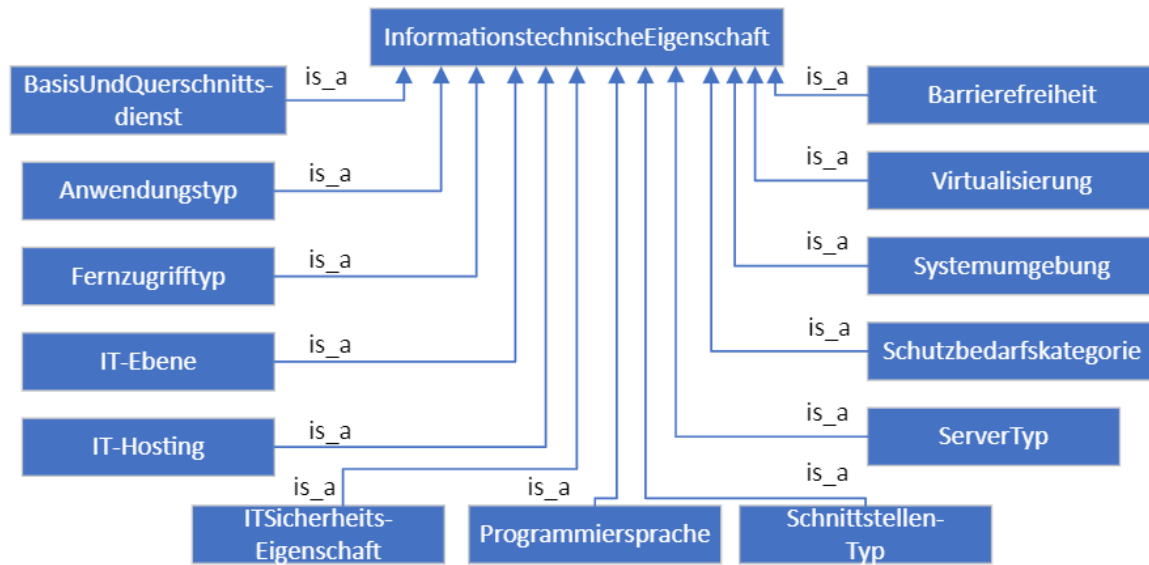


Figure 10: Subclasses of the class *InformationstechnischeEigenschaft*

The information technology properties are used to represent safety-critical IT systems with regard to their possible technical properties. Table 9 below describes the respective subclasses of information technology properties.

Class	Description	Subclass of
BasisUndQuerschnitts-dienste	The class <i>BasisUndQuerschnitts dienste</i> serves as the upper class of all basic and cross-sectional services in a safety-critical IT system. Services that form a common, cross-sectional basis for other services, e.g., specialist services, services based on them, are not directly assigned to any individual specialist task.	Informationstechnische Eigenschaft
Anwendungstyp	The class includes various application types that are currently available in information technology.	Informationstechnische Eigenschaft
Fernzugriffstyp	The class represents the superclass for the various types of remote access to IT systems. Remote access is understood to mean external access (outside the network) to the safety-critical IT system.	Informationstechnische Eigenschaft

IT-Ebene	The class represents the superclass for the IT levels of an IT system.	Informationstechnische Eigenschaft
IT-Hosting	The class includes all types of IT hosting for safety-critical IT systems.	Informationstechnische Eigenschaft
ITSicherheits Eigenschaft	The class includes all types of IT security properties of a security-critical safety-critical IT system. The IT security properties represent elementary properties of a security-critical IT system in order to guarantee the protection objectives of the KRITIS requirement. IT security properties are characterized by the combination of several individual IT security properties	Informationstechnische Eigenschaft
Programmiersprache	The class represents all types of programming languages that can be used in safety-critical IT projects.	Informationstechnische Eigenschaft
Schnittstellen Typ	The class subsumes all IT interface types that could be used in safety-critical IT projects. The interface types include open and non-open standards. IT interface types are data-oriented transitions between programs or services via which data is exchanged.	Informationstechnische Eigenschaft
ServerTyp	The class <i>ServerTyp</i> subsumes various server types that can be used from an application perspective for the implementation of a safety-critical IT system. The class does not include the server type from a hardware view.	Informationstechnische Eigenschaft
Schutzbedarfskategorie	The class serves as a superclass for the protection requirement categories. The protection requirement is used to classify a security-critical IT system.	Informationstechnische Eigenschaft
Systemumgebung	The class represents all types of system environments that are used for safety-critical IT systems.	Informationstechnische Eigenschaft

Virtualisierung	The class serves as a superclass for all types of virtualization of IT hardware components.	Informationstechnische Eigenschaft
Barrierefreiheit	The class serves as a superclass for the accessibility properties of an IT application.	Informationstechnische Eigenschaft

Table 9: Description of the subclasses of the class *InformationstechnischeEigenschaft*

The class *VergabeverfahrensArt* and its subclasses represent all types of public procurement procedures. Figure 11 below shows the class *VergabeverfahrensArt* and the subclasses.

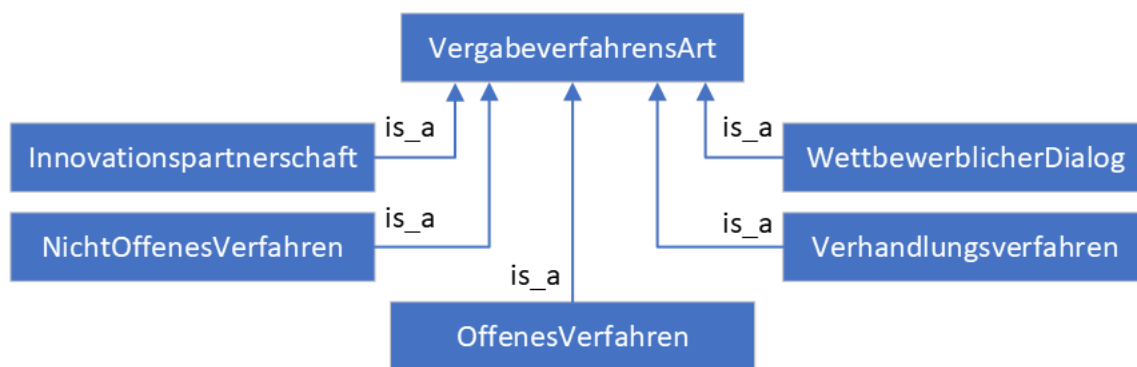


Figure 11: Subclasses of the class *VergabeverfahrensArt*

Public procurement procedures are highly committed to openness, transparency, and the principle of fairness, as public funds are used for procurement in the public interest. A lack of restriction and transparency can make a procurement procedure appear arbitrary and give rise to the suspicion of influence.

An award procedure in the context of security-critical IT projects poses a particular challenge in terms of transparency and legal certainty. The awarding of a security-critical IT system can take one to two years from start to finish. This does not take into account a possible lawsuit by an unsuccessful bidder, which can lead to a further delay. This means that an award procedure also influences the safety-critical IT project.

Figure 12 below illustrates the award steps provided for the respective award procedure types. In security-critical IT projects, the award documents may be subject to a confidentiality level, meaning that all said relevant documents are subject to a confidentiality classification.

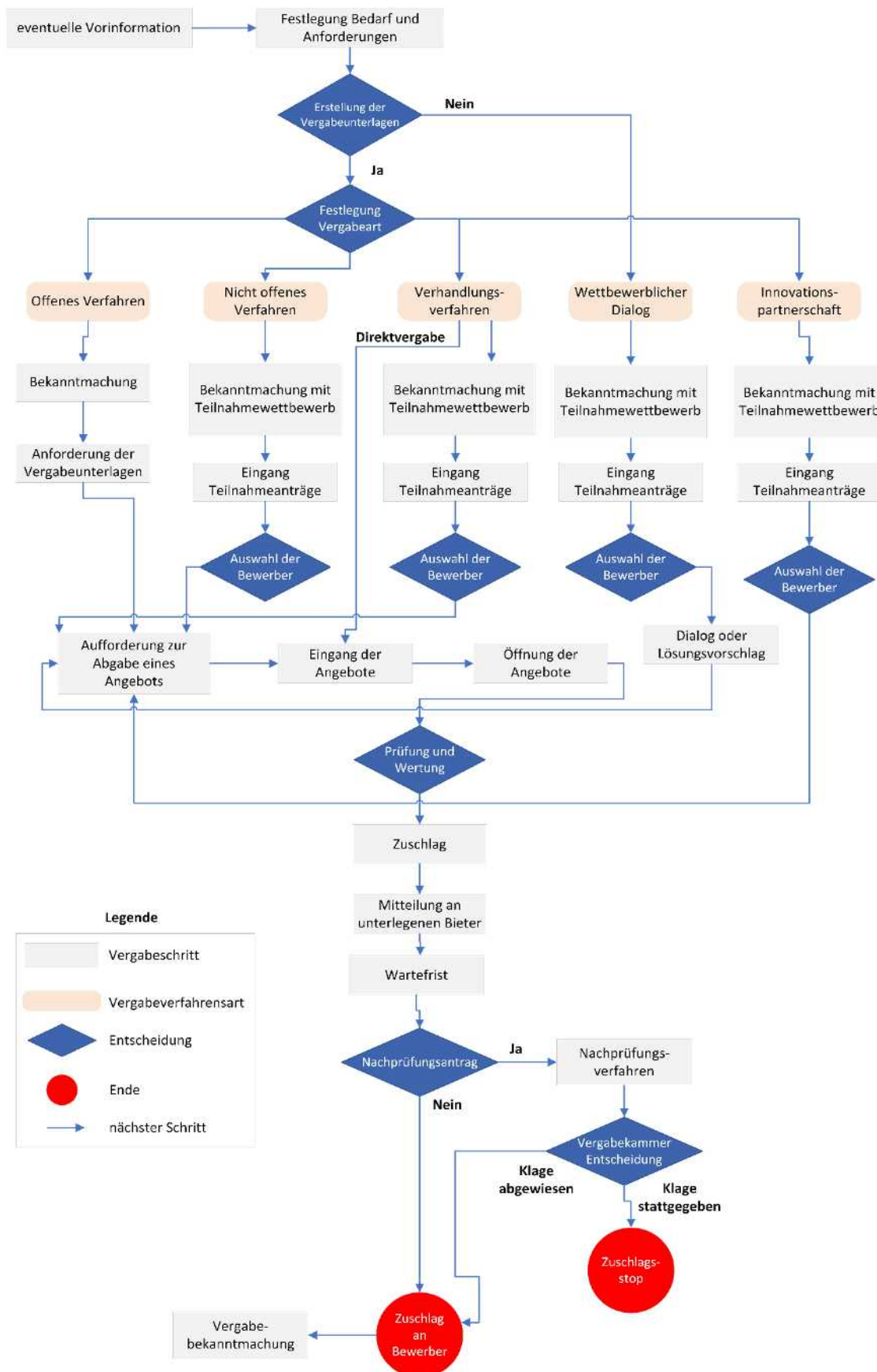


Figure 12: Award procedure

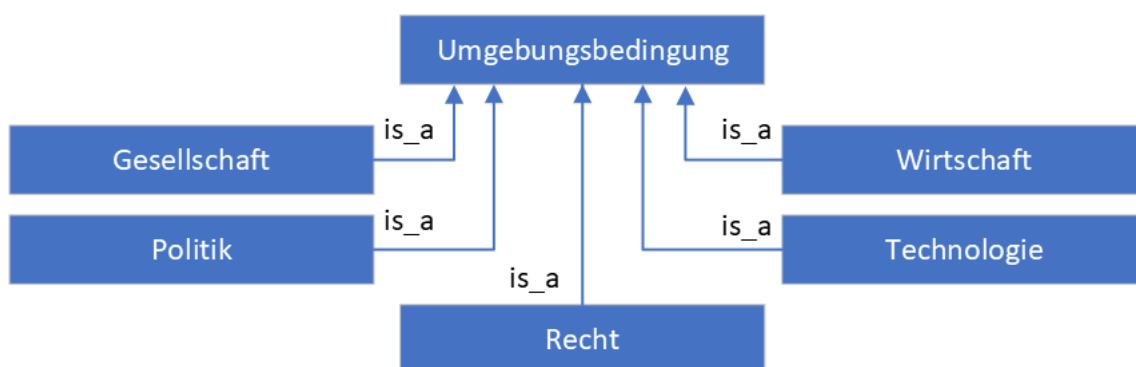
Table 10 explains the subclasses of the class *VergabeverfahrensArt*.

Class	Description	Subclass of
Innovationspartnerschaft	The class <i>Innovationspartnerschaft</i> represents the type of award procedure for an innovation partnership. The innovation partnership is a procedure for the development of innovative products that are not yet available on the market for the subsequent acquisition of the resulting services. An innovation partnership is a type of award procedure that was added in 2016.	VergabeverfahrensArt
NichtOffenesVerfahren	The class <i>NichtOffenesVerfahren</i> provides the linguistic means of expression for the non-open award procedure. A restricted award procedure is one in which the contracting authority, following a prior public invitation to participate (call for competition), selects a limited number of companies on the basis of objective, transparent, and non-discriminatory criteria, which it invites to submit tenders.	VergabeverfahrensArt
OffenesVerfahren	The class <i>OffenesVerfahren</i> represents the open procurement procedure. This is a type of procurement procedure in which the contracting authority publicly invites an unlimited number of companies to submit bids.	VergabeverfahrensArt

Verhandlungsverfahren	The class <i>Verhandlungsverfahren</i> represents the negotiated procedure. This is a type of procurement procedure in which the contracting authority approaches selected companies, with or without a call for competition, in order to negotiate the tenders with one or more of these companies.	VergabeverfahrensArt
WettbewerblicherDialog	The class <i>Wettbewerblicher Dialog</i> provides the linguistic means of expression for describing the competitive dialogue type of award procedure, for the award of public contracts with the aim of identifying and determining the means by which the contracting authority's needs can best be met. This contracting authority conducts a dialogue with the selected companies to discuss all aspects of the contract awarding.	VergabeverfahrensArt

Table 10: Description of the subclasses of the class *VergabeverfahrensArt*

The environmental conditions influence a safety-critical IT project in various ways, which is expressed by the following subclasses. Figure 13 below shows the class *Umgebungsbedingung* with its subclasses.

Figure 13: Subclasses of the class *Umgebungsbedingung*

The importance of environmental conditions can sometimes be found as framework conditions in service descriptions of safety-critical IT projects, making the potential provider aware of said conditions. The subdivision into the following subclasses is based on the PEST concept. The term PEST is made up of the four English terms “Political”, “Economic”, “Sociological”, and “Technological”. For the following analysis, the four dimensions to include a fifth “Law”. This is because we are setting out to classify security-critical IT projects in the area of public procurement, meaning that public procurement law plays a key role. Table 11 represents the class *Umgebungsbedingung* with its subclasses.

Class	Description	Subclass of
Gesellschaft	The class <i>Gesellschaft</i> subsumes all environmental conditions that are influenced by society. Societal environmental conditions can be structural characteristics (population structure, income structure, proportion of city dwellers, educational background, etc.) and associated trends (e.g., demographic change, increased need for security). Demands on products or on companies themselves are also summarized under social environmental conditions (increasing environmental and health awareness, individualization, digital sovereignty, citizens’ sensitivity to data protection).	Umgebungsbedingung
Politik	The class <i>Politik</i> subsumes the environmental conditions of politics. The environmental factor “politics” comprises the influencing factor of politics as the highest responsible party for the procurement of a safety-critical IT system. The political framework conditions depend on the respective legislative period with the associated political agendas and are independent of a project duration. In this case, the “political” factor does not include the legal framework conditions, which are expressed in the following “legal” factor.	Umgebungsbedingung

Recht	The class <i>Recht</i> provides the linguistic means of expression for the legal framework conditions. The legal framework is defined at the municipal, regional, national, and supranational level. They comprise the legal framework conditions for the contracting authority and the contractor. These include regulations of the corporate constitution, public procurement law, obligations under the Public Procurement Act (VerpflG), declaration of commitment to comply with the ILO core labor standards, obligation to handle classified information of the classification level VS-NfD, data protection regulations, taxation, liability conditions according to EVB-IT, no-spy regulation including the technical no-spy clause, security regulations according to BSI, KRITIS regulation, online accessibility law, and accessibility. Security-critical IT projects face the challenge of having to implement existing legal requirements at the federal and state level.	Umgebungsbedingung
Technologie	The class <i>Technologie</i> subsumes the technological environmental conditions. Technological developments influence IT projects in a variety of ways. Technology trends can lead to new technologies being developed or existing technologies being discontinued. The aspect of technology security plays an important role in security-critical IT projects. The technology used in security-critical IT projects must meet the highest requirements in terms of operational security and future-proofing. The ability of authorities and organizations with security tasks must be guaranteed at all times. Larger consulting groups publish an annual presentation of the IT trends that should be taken into account in the technological implementation of IT projects.	Umgebungsbedingung
Wirtschaft	The class <i>Wirtschaft</i> is the upper class for the economy's ambient conditions. Economic environmental conditions play an important role in public procurement. The contracting authorities (federal/state/local authorities) have different economic framework conditions. The federal states or municipalities themselves also have different framework conditions at the economic level. For example, one federal state may be in a better economic position than another. This can be reflected in public contracts.	Umgebungsbedingung

Table 11: Description of the subclasses of the class *Umgebungsbedingung*

Figure 14 shows the upper class *DerivativeQuantitativeEigenschaft* with its subclasses.

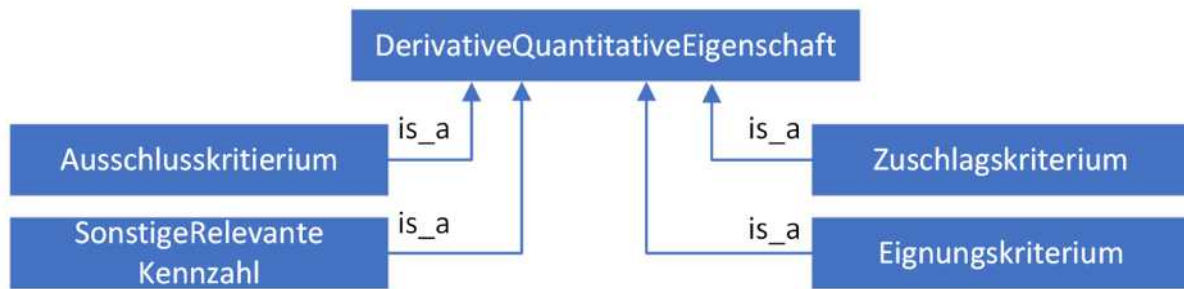


Figure 14: Subclasses of the class *DerivativeQuantitativeEigenschaft*

The derivative quantitative properties represent key figures that enable a derived evaluation. The class *DerivativeQuantitativeEigenschaft* can be easily explained using the suitability, award, and exclusion criteria from a public procurement procedure. These criteria can be used in a public procurement procedure to derive whether a bidder can, for example, successfully pass a competition or be excluded from a procurement procedure because it does not meet the suitability criteria.

The four subclasses *Ausschlusskriterium*, *Eignungskriterium*, *SonstigeRelevanteKennzahl*, und *Zuschlagskriterium* are discussed below.

Class	Description	Subclass of
Ausschlusskriterium	The class <i>Ausschlusskriterium</i> represents the upper class for exclusion criteria in public procurement procedures. Generally applicable exclusion criteria leave the contracting authority no room for discretion. In addition to generally applicable exclusion criteria, the notice of a request to participate or the notice of a call for tenders also contains other substantive exclusion criteria that must be taken into account when a bidder is processing a tender.	DerivativeQuantitative Eigenschaft
Sonstige Relevante Kennzahl	The class <i>SonstigeRelevanteKennzahl</i> subsumes the other relevant key figures as a superclass.	DerivativeQuantitative Eigenschaft

Zuschlagskriterium	The class <i>Zuschlagskriterium</i> represents the upper class of award criteria in the context of a contract award procedure. The award criteria are selection criteria used by the contracting authority to award the contract. The contracting authorities are obliged to disclose the award criteria, which may also have various sub-categories, in the tender documents for the invitation to tender. The award criteria may have different weightings in the evaluation. The weighting must also be disclosed. Changing the award criteria during an award procedure is not permitted. The award criteria express to which characteristics the contracting authority essentially attaches importance and what serves as the basis for evaluating the tender. It is particularly important for potential contractors to fulfill the award criteria.	DerivativeQuantitative Eigenschaft
Eignungskriterium	The class <i>Eignungskriterium</i> serves as the upper class of all suitability criteria in a public procurement procedure. According to the ARC requirements, a potential bidder is deemed suitable if it fulfills the suitability criteria specified in the award documents. The suitability criteria must be distinguished from the award criteria. The suitability criteria indicate whether a potential bidder is suitable to submit an offer. These suitability criteria may only include the following: • License and qualification to practice the profession • Financial and economic performance • Professional and technical performance The suitability criteria must be listed in full in the contract award notice. They must be proportionate and serve exclusively to identify the companies that are suitable to provide the requested services. As a rule, suitability criteria are defined in participation competitions in order to identify suitable participants from the point of view of the contracting authority for the invitation to tender.	DerivativeQuantitative Eigenschaft

Table 12: Description of the subclasses of the class *DerivativeQuantitativeEigenschaft*

Figure 15 below shows the class *OrginäreQuantitativeEigenschaft* with the respective subclasses.

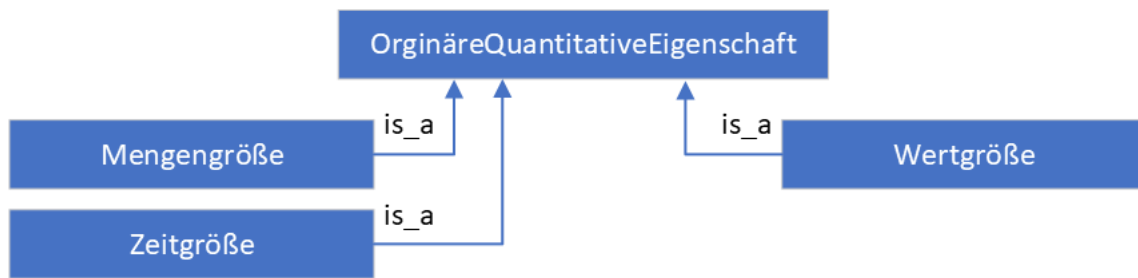


Figure 15: Subclasses of the class *OrginäreQuantitativeEigenschaft*

The class *OrginäreQuantitativeEigenschaft* represents measurable units of quantity. The main difference from the class *DerivativeQuantitativeEigenschaft* is that the original quantitative property is understood as an assessment-free general property, for example as quantity or time quantities.

Class	Description	Subclass of
Mengengröße	The class <i>Mengengröße</i> serves as the superclass of all quantity quantities. The quantity quantities specify a measurable unit or quantity of a defined property that is not based on a time or value quantity.	OrginäreQuantitative Eigenschaft
Zeitgröße	The class <i>Zeitgröße</i> serves as the superclass of all time variables. The time variable describes a specific point in time, a specific date, or a time interval.	OrginäreQuantitative Eigenschaft
Wertgröße	The class <i>Wertgröße</i> serves as the superclass of all value variables. The value variables describe monetary values.	OrginäreQuantitative Eigenschaft

Table 13: Description of the subclasses of the class *OrginäreQuantitativeEigenschaft*

Figure 16 shows the class *ITSicherheitsEigenschaft* with the associated subclasses.

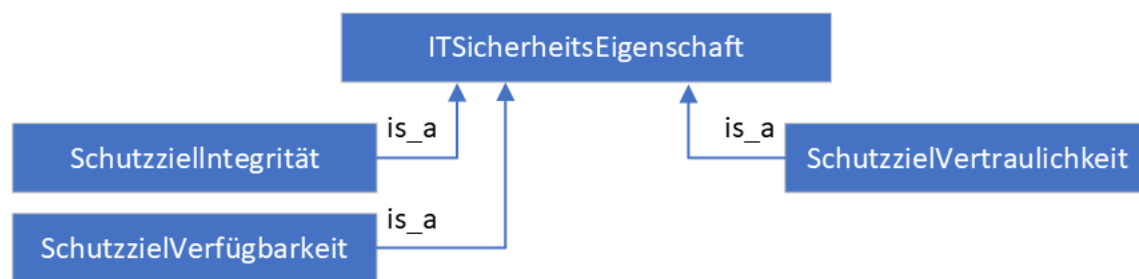


Figure 16: Subclasses of the class *ITSicherheitsEigenschaft*

IT security features are of fundamental importance for a safety-critical IT system and are included in the relevant performance specifications for safety-critical IT systems as a mandatory requirement.

Protection goals are suitable for defining IT security properties. In the relevant international specialist literature, these protection goals are referred to as the CIA triad (Confidentiality, Integrity, and Availability) and are regarded as fundamental characteristics of IT security; cf. GONIWADA (2022), pp. 374–375; LIEDTKE (2022), p. 19. The protection goals of integrity, availability, and confidentiality are fundamental goals of IT security. It is important to distinguish between the protection goals, which are defined below as IT security properties, and the basic threats. The basic threats describe potential dangers to the security of security-critical IT systems, such as data loss, data theft, or unauthorized access to data. A protection goal can therefore be understood as one that is to be achieved through security measures (thus as an IT security property), while the basic threat represents one that jeopardizes achieving the protection goal. It is therefore important that both the basic threats and the protection goals are taken into account when planning security measures. The identified need for protection is difficult to quantify and is therefore limited to a qualitative assessment.

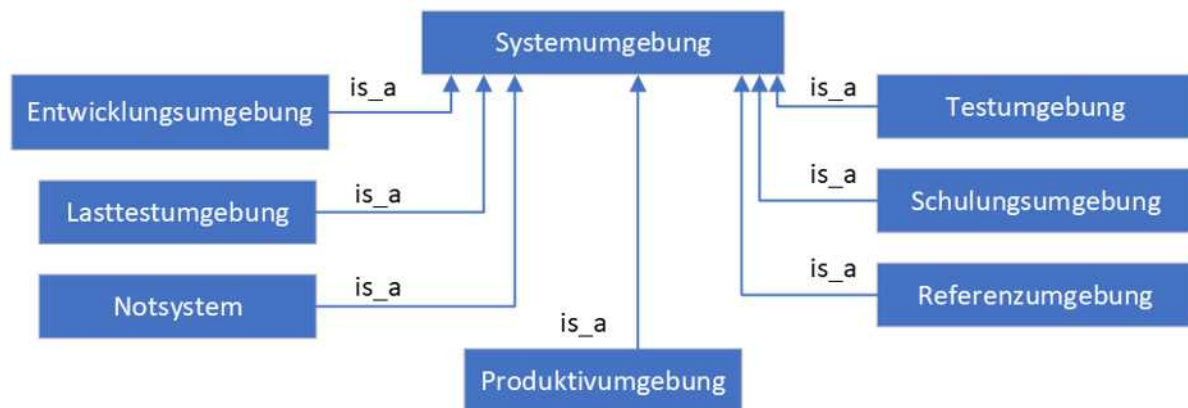
Table 14 below shows the subclasses of the class *ITSicherheitsEigenschaft*.

Class	Description	Subclass of
Schutzziel Integrität	The class <i>SchutzzielIntegrität</i> serves as a superclass for all properties relating to the protection objective of integrity. According to IT baseline protection, the “integrity” protection objective refers to the consistent functioning of IT systems and the completeness and accuracy of data.	ITSicherheits Eigenschaft

Schutzziel Verfügbarkeit	As a superclass, the class <i>SchutzzielVerfügbarkeit</i> contains all properties relating to the availability protection objective. The “availability” protection objective defines the degree to which IT systems, IT applications, IT networks and data are available to a user and can be used without restriction.	ITSicherheits Eigenschaft
Schutzziel Vertraulichkeit	The class <i>SchutzzielVertraulichkeit</i> serves as a superclass for all properties relating to the confidentiality protection objective. This objective defines the protection against unauthorized disclosure of confidential data. Confidential data may only be accessible to authorized persons in the permitted manner.	ITSicherheits Eigenschaft

Table 14: Description of the subclasses of the class *ITSicherheitsEigenschaft*

Figure 17 illustrates the class *Systemumgebung* with the associated subclasses.

Figure 17: Subclasses of the class *Systemumgebung*

System environments are of central importance for safety-critical IT systems, as they are crucial for fail safety and for testing software and hardware. An adaptation to a productive environment in the form of a software, configuration, or hardware adaptation is tested on various system environments beforehand. As soon as it is ensured that the safety-critical IT system is unlikely to fail, the adaptation to the production environment can take place. The purpose of running through the various system environments is to identify possible technical risks, such as those that may arise from software or hardware, on the various system environments at an early stage in order to rule them out for the

production environment. The purpose of running through the various system environments is to identify possible technical risks, such as those caused by software or hardware, on the various system environments at an early stage in order to rule them out for the production environment. The class *Systemumgebung* is differentiated into the subclasses *Testumgebung*, *Lasttestumgebung*, *Notsystem*, *Referenzumgebung*, and *Produktivumgebung* and explained in more detail in Table 15 below as a subclass of the class *Systemumgebung*.

Class	Description	Subclass of
Entwicklungs- umgebung	The class <i>Entwicklungsumgebung</i> serves as a superclass for all development environments. A development environment is a system environment that is mainly used for the development of software. The software developers of an application use this system environment to carry out development tests, for example.	Systemumgebung
Lasttest- umgebung	The class <i>Lasttestumgebung</i> serves as a superclass for all load test environments. A load test environment is a system environment that is set up specifically for load tests. The aim of load tests is to determine the maximum utilization of an IT system and to check the system's availability under load. Attention is paid to performance and reliability not only for conventional IT systems, but also for safety-critical IT systems. Regular load tests are necessary in order to determine the utilization limit of a safety-critical IT system, even with constant further development and adaptation. As load tests can severely impair the system's performance, they should not be carried out in a productive environment. Instead, they are carried out in the load test environment so as not to jeopardize the productive operation of the system.	Systemumgebung
Notsystem	The class <i>Notsystem</i> serves as the superclass for all emergency systems. An emergency system is a system environment that is used when the productive environment has failed. Activation of the emergency system can be triggered automatically or manually. An emergency system is a fallback environment for the productive operation of a security-critical IT system.	Systemumgebung

Produktiv umgebung	The class <i>Produktivumgebung</i> serves as a superclass for all productive environments. The production environment is the system environment actively used by the users of a safety-critical IT system. The production environment is the most important system environment, as it is where the productive operation of the safety-critical IT system takes place. The production environment's failure would result in a system failure.	Systemumgebung
Referenz umgebung	The class <i>Referenzumgebung</i> serves as a superclass for all reference environments. The reference environment is the system environment that serves as a reference for the production environment. The environment is used to carry out software tests on a production-like environment before software is installed on the production environment. As a rule, the reference environment is similar or identical in structure to a production environment.	Systemumgebung
Schulungs umgebung	The class <i>Schulungsumgebung</i> serves as a superclass for all training environments. The training environment is the system environment in which the user groups (e.g., administrators) are trained as part of training courses.	Systemumgebung
Testumgebung	The class <i>Testumgebung</i> serves as a superclass for all test environments. The test environment is the system environment in which comprehensive software, data, and hardware tests take place.	Systemumgebung

Table 15: Description of the subclasses of the class *Systemumgebung*

Figure 18 below shows the class *Virtualisierung* with the associated subclasses.

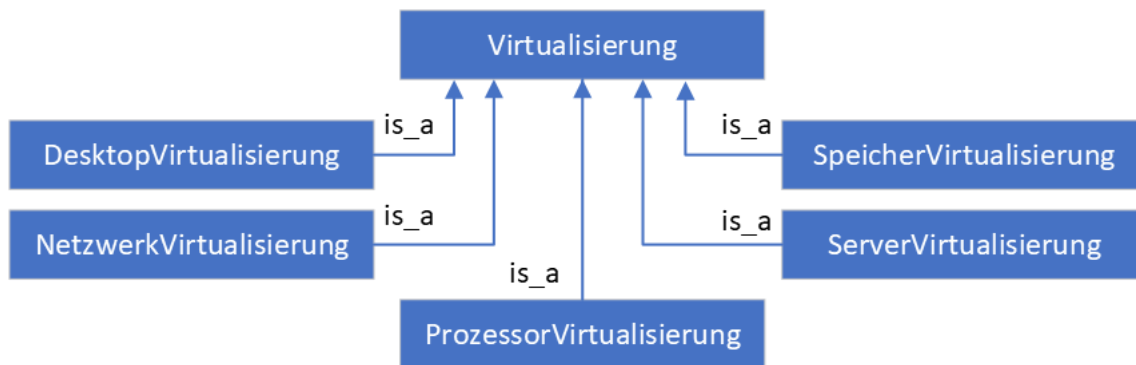


Figure 18: Subclass of the class *Virtualisierung*

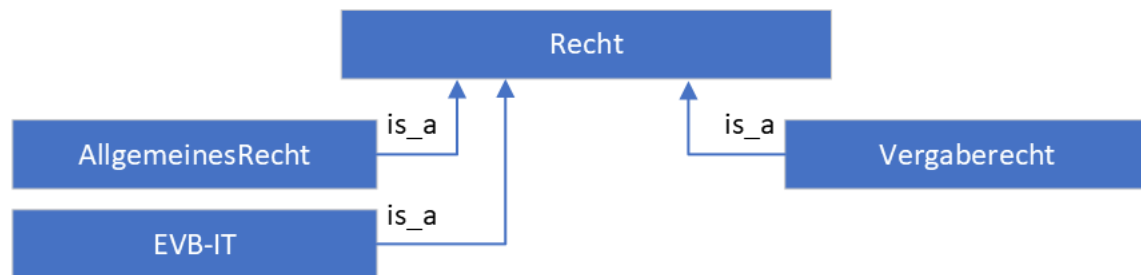
The virtualization of hardware components is a common way of providing technical resources in information technology. Table 16 below explains the subclasses of the class *Virtualisierung*, which provide the linguistic means of expression to express the different virtualization concepts.

Class	Description	Subclass of
Desktop Virtualisierung	The class <i>DesktopVirtualisierung</i> serves as a superclass for all types of desktop virtualization. Desktop virtualization refers to a virtualization concept in which a desktop client is provided virtually.	Virtualisierung
Netzwerk Virtualisierung	The class <i>NetzwerkVirtualisierung</i> serves as a superclass for all types of network virtualization. Network virtualization is a virtualization concept in which a network is provided virtually.	Virtualisierung
Prozessor Virtualisierung	The class <i>ProzessorVirtualisierung</i> serves as a superclass for all types of processor virtualization. Processor virtualization refers to a virtualization concept in which a processor is provided virtually.	Virtualisierung
Server Virtualisierung	The class <i>ServerVirtualisierung</i> serves as a superclass for all types of server virtualization. Server virtualization is a virtualization concept in which a server is provided virtually.	Virtualisierung

Speicher Virtualisierung	The class <i>SpeicherVirtualisierung</i> serves as a superclass for all types of storage virtualization. Storage virtualization refers to a virtualization concept in which storage is provided virtually.	Virtualisierung
-----------------------------	---	-----------------

Table16: Description of the subclasses of the class *Virtualisierung*

Figure 19 below shows the class *Recht* with the associated subclasses.

Figure 19: Subclasses of the class *Recht*

The subclasses of the class *Recht* represent different central legal issues for security-critical IT projects. These include general law, the EVB-IT (Supplementary Contract Terms for the Procurement of IT Services), and procurement law. The subdivision is due to the fact that legal issues for a security-critical IT project can be distinguished between general legislation, the contract law for the procurement of IT services of the Federal Republic of Germany, and public procurement law. The EVB-IT and public procurement law are legal frameworks that have an impact on the public procurement of IT services and differ from general law, which can be understood as a basic framework.

Contracts for the procurement of safety-critical IT systems require comprehensive regulation in various areas due to the procurement's complexity of the procurement and its high significance for the company, such as the description of both parties' performance obligations and the definition of liability conditions in the event of malfunctions or failures of such systems. The statutory provisions on the law of obligations in the German Civil Code (BGB) alone do not sufficiently meet this requirement. Although they provide indications for a possible contract, they fail to adequately take IT-specific aspects into account.

As a result, framework conditions must be created to supplement the aforementioned existing legal bases in order to place such complex projects on a contractual footing.

The Federal Ministry of the Interior has therefore drawn up these supplementary contract terms for information technology (EVB-IT) with regard to the procurement of public IT services. The EVB-IT subclass subsumes the various EVB-IT contract types that can be used for different areas of application.

We have already touched on public procurement law in the class *VergabeverfahrensArt*, as security-critical IT projects intended for use by public authorities and organizations with security tasks (BOS) are subject to public procurement law. The procurement procedure can be understood as the application of public procurement law. With its regulations, public procurement law defines how the federal government, federal states, and local authorities must proceed in order to purchase goods on the market or to commission construction and services. It is intended to ensure that the contracting authorities' budget funds are used economically and in a competitive, transparent, and non-discriminatory award procedure in order to give preference to the most economical offer in terms of value for money.

Table 17 below explains the subclasses of the class *Recht*.

Class	Description	Subclass of
Allgemeines Recht	The class <i>AllgemeinesRecht</i> includes all generally applicable laws.	Recht
EVB-IT	The class <i>EVB-IT</i> is the upper class for the EVB-IT's different contractual conditions.	Recht
Vergaberecht	The class <i>Vergaberecht</i> represents the upper class of public procurement law, which subsumes the current and old public procurement law of the Federal Republic of Germany.	Recht

Table 17: Description of the subclasses of the class *Recht*

Figure 20 below shows the subclasses of the class *Zuschlagskriterium*.

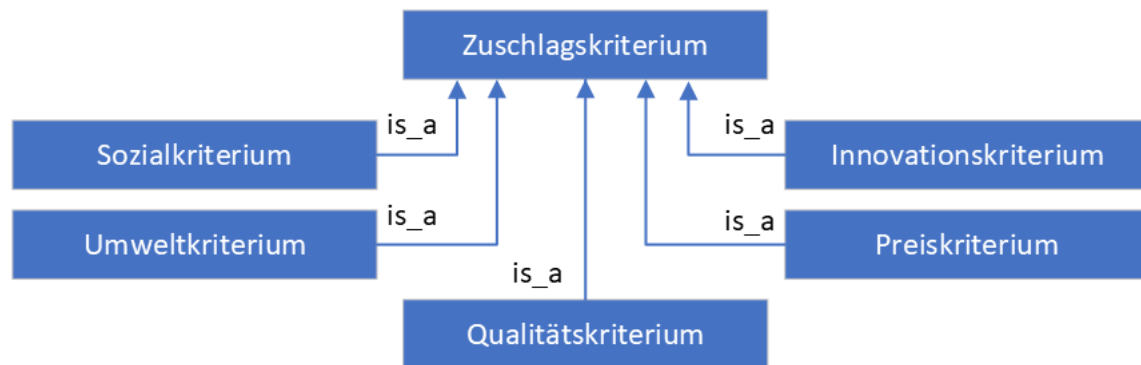


Figure 20: Subclasses of the class *Zuschlagskriterium*

The contract is awarded in accordance with Section 127 GWB (Act against Restraints of Competition) to the most economically advantageous tender, who is usually determined on the basis of a price-performance ratio. Section 58 VgV (Public Procurement Regulation) “Award and Award Criteria” lists qualitative, innovative, environmental, and social criteria as award criteria in addition to price. This also justifies the selection of the classes explained in Table 18 as subclasses of the class *Zuschlagskriterien*. The award criteria are of central importance for the award of a public contract and are therefore also the subject of various applications for review. The award criteria must be related to the contract’s subject matter.

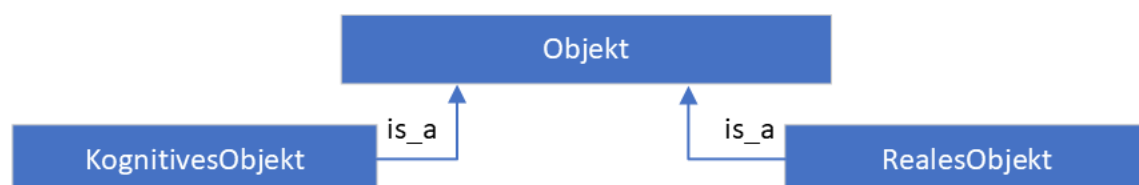
Class	Description	Subclass of
Sozialkriterium	The class <i>Sozialkriterium</i> represents the upper class for all award criteria that take social criteria into account. The Public Procurement Ordinance (VgV) describes social criteria as the “Zugänglichkeit der Leistung insbesondere für Menschen mit Behinderungen” and “Kriterien, die bestimmte Aspekte des gesellschaftlichen Zusammenlebens betreffen”.	Zuschlagskriterium
Umweltkriterium	The class <i>Umweltkriterium</i> represents the upper class for all award criteria based on environmental criteria. Environmental award criteria are those linked to service characteristics that reflect the service’s effect on the environment.	Zuschlagskriterium

Qualitätskriterium	The class <i>Qualitätskriterium</i> represents the upper class for all award criteria that are based on quality-related criteria. Quality criteria are those that can influence the quality of the tendered service. The VgV describes, among other things, that the qualifications and experience of the personnel responsible for the contract's execution can have an influence on said execution's quality. Furthermore, the availability of the service and technical assistance as well as delivery conditions such as delivery date, delivery procedure, and delivery and execution deadlines are also to be included under quality criteria.	Zuschlagskriterium
Preis-kriterium	The class <i>Preiskriterium</i> is the superordinate class for all award criteria that are based on price-related criteria. Price-related award criteria are those linked to a service's price.	Zuschlagskriterium
Innovationskriterium	The class <i>Innovationskriterium</i> is the superordinate class for all award criteria that are based on innovation-related criteria.	Zuschlagskriterium

Table 18: Description of the subclasses of the class *Zuschlagskriterium*

In addition to the class *Eigenschaft*, the class *Objekt* is a central class on the first hierarchy level of the safety-critical IT project ontology—a level we already defined in the underlying PM domain ontology. Selected subclasses of the class *Objekt* are described in more detail below.

The class *Objekt* is divided into the subclasses *KognitivesObjekt* and *RealesObjekt* by the PM domain ontology used, as illustrated in Figure 21 below.

Figure 21: Subclasses of the class *Objekt*

The distinction between the subclasses *KognitivesObjekt* and *RealesObjekt* is described in the security-critical IT project ontology as a differentiation between objects of thought

and objects of experience. Cognitive objects are objects that are created by the “inner perception” of any actor, regardless of whether the objects exist in reality. Real objects, on the other hand, are the content of experiential processes. The real objects exist in the respective conceptualized domain and can be observed.

Table 19 below explains the subclasses of the class *Objekt*.

Class	Description	Subclass of
KognitivesObjekt	The class <i>KognitivesObjekt</i> is the superclass of all cognitive objects. The subclasses of the class <i>KognitivesObjekt</i> represent objects of thought processes that arise through “inner perception.” The cognitive objects do not necessarily have to exist in reality.	Objekt
RealesObjekt	The class <i>RealesObjekt</i> is the superclass of all real objects. The subclasses of the class <i>RealesObjekt</i> represent objects of experience processes that arise through “external perception.” Real objects exist in the constructed domain and can be observed by actors using their senses.	Objekt

Table 19: Description of the subclasses of the class *Objekt*

Figure 22 below shows the class *System* with the associated subclasses.

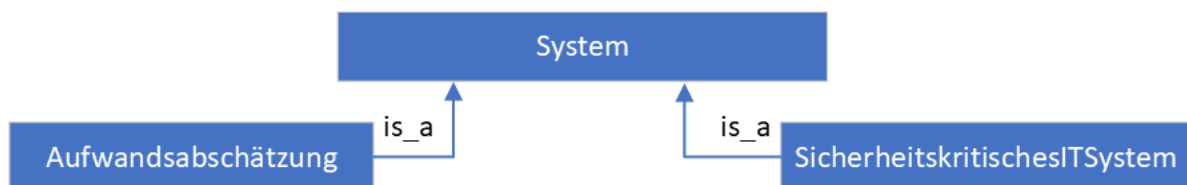


Figure 22: Subclasses of the class *System*

Table 20 below explains the subclasses of the class *System*.

Class	Description	Subclass of
Aufwandsabschätzung	The class <i>Aufwandsabschätzung</i> represents the superclass for all types of effort estimation. It covers different types of effort estimation.	System
Sicherheitskritisches ITSystem	The class <i>SicherheitskritischesITSystem</i> is the upper class for all types of safety-critical IT systems.	System

Table 20: Description of the subclasses of the class *System*

Figure 23 below shows the class *Urteil* with the associated subclasses.

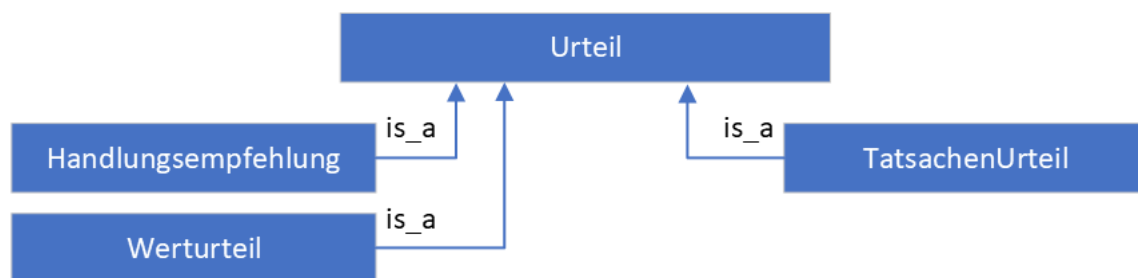


Figure 23: Subclasses of the class *Urteil*

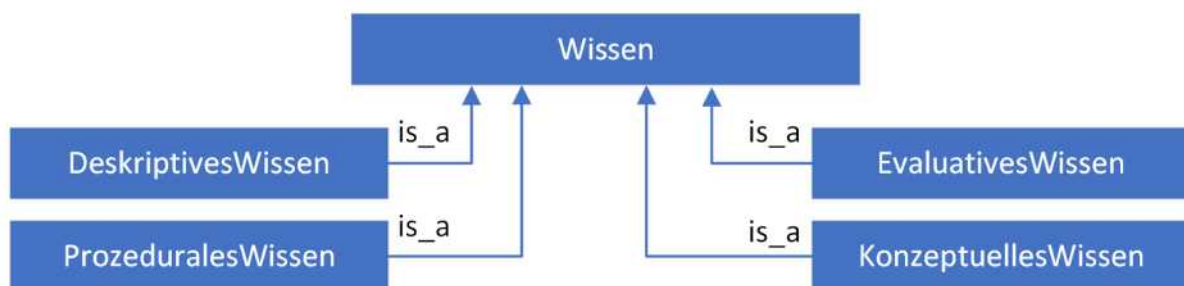
The class *Urteil* is subdivided into the classes *Handlungsempfehlung*, *Werturteil*, and *TatsachenUrteil*. Table 21 below explains the subclasses of the class *Urteil*.

Class	Description	Subclass of
Handlungsempfehlung	The class <i>Handlungsempfehlung</i> is the superclass for all types of recommended actions. Recommendations for action are characterized as suggestions for action plans that do not necessarily have to be adhered to. They serve as recommendations for a course of action. Recommendations for action are published by software providers or public institutions, for example, to support a course of action.	Urteil

Werturteil	The class <i>Werturteil</i> is the superclass for all types of value judgments. Value judgements are characterized as personal statements in which a fact is described positively or negatively. In contrast to statements of fact, value judgments are not empirically verifiable.	Urteil
Tatsachen Urteil	The class <i>TatsachenUrteil</i> is the superclass for all types of court judgments.	Urteil

Table 21: Description of the subclasses of the class *Urteil*

Figure 24 below illustrates the class *Wissen* with the associated subclasses.

Figure 24: Subclasses of the class *Wissen*

The classes *DeskriptivesWissen*, *EvaluativesWissen*, *KonzeptuellesWissen*, and *ProzeduralesWissen* are used for the safety-critical IT project ontology. Table 22 below explains the subclasses mentioned.

Class	Description	Subclass of
Deskriptives Wissen	The class <i>DeskriptivesWissen</i> is the superclass for all types of descriptive knowledge. “Descriptive knowledge” is defined as the knowledge of facts expressed through theories, concepts, principles, schemes, and ideas.	Wissen
Prozedurales Wissen	The class <i>ProzeduralesWissen</i> is the superclass for all types of procedural knowledge. “Procedural knowledge” describes the practically applicable action knowledge and the ability to interlink declarative knowledge and apply it in action sequences.	Wissen

Evaluatives Wissen	The class <i>EvaluativesWissen</i> is the superclass for all types of evaluative knowledge. “Evaluative knowledge” represents knowledge that is available in the form of an evaluation.	Wissen
Konzeptuelles Wissen	The class <i>KonzeptuellesWissen</i> is the superclass for all types of conceptual knowledge. “Conceptual knowledge” is the basis for an in-depth understanding of subject-specific content. It is defined as the understanding of concepts that influence a domain and can be placed in a context. Conceptual knowledge comprises explanatory mechanisms for very specific facts that need to be linked together.	Wissen

Table 22: Description of the subclasses of the class *Wissen*

Figure 26 below shows the class *DeskriptivesWissen* with the associated subclasses.

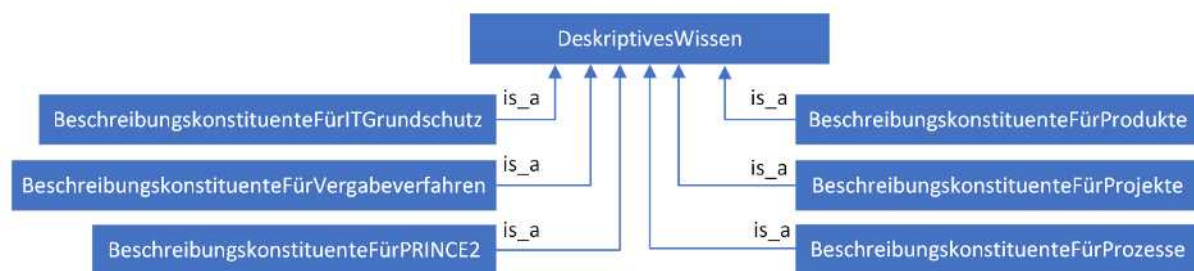
Figure 25: Subclasses of the class *DeskriptivesWissen*

Table 23 below explains the subclasses of the class *DeskriptivesWissen*.

Class	Description	Subclass of
Beschreibungs konstituente FürITGrundschutz	The class <i>BeschreibungskonstituenteFürITGrundschutz</i> serves as a superclass that provides the linguistic means of expression for expressing IT-Grundschutz.	Deskriptives Wissen
Beschreibungs konstituente FürVergabeverfahren	The class <i>BeschreibungskonstituenteFürVergabeverfahren</i> is the superclass that provides the linguistic means of expression to express the awarding steps.	Deskriptives Wissen

Beschreibungs konstituente- FürPRINCE2	The class <i>BeschreibungskonstituenteFürPRINCE2</i> acts as a superordinate class that encompasses all types of description constituents of the PRINCE2 project management method. In this respect, the class provides the linguistic means of expression required to describe the defined terms in the PRINCE2 project management method.	Deskriptives Wissen
Beschreibungs konstituenteFür Produkte	The class <i>BeschreibungskonstituentenFürProdukte</i> serves as a superclass for all types of description constituents that are relevant for products. It provides linguistic means of expression for the requirements for real goods (expressed with the class <i>Realgut</i>).	Deskriptives Wissen
Beschreibungs konstituenteFür Projekte	The class <i>BeschreibungskonstituenteFürProjekte</i> is the superclass for all types of description constituents in the context of projects. It provides the linguistic means of expression for projects, thus enables a uniform and standardized description.	Deskriptives Wissen
Beschreibungs konstituenteFür Prozesse	The class <i>BeschreibungskonstituenteFürProzesse</i> is the superclass for all types of descriptive constituents in the context of processes. The class provides linguistic means of expression for actions, thus enable a comprehensive description of processes.	Deskriptives Wissen

Table 23: Description of the subclasses of the class *DeskriptivesWissen*

Figure 26 below shows the class *EvaluativesWissen* with the associated subclasses.

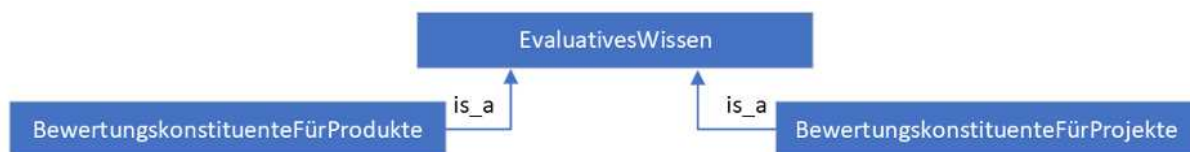
Figure 26: Subclasses of the class *EvaluativesWissen*

Table 24 below explains the subclasses of the class *EvaluativesWissen*.

Class	Description	Subclass of
BewertungskonstituenteFürProdukte	The class <i>BewertungskonstituenteFürProdukte</i> functions as a superordinate class for all types of evaluations in the context of products. This class provides linguistic means of expression for evaluating the fulfillment of requirements.	EvaluativesWissen
BewertungskonstituenteFürProjekte	The class <i>BewertungskonstituenteFürProjekte</i> functions as a superclass for all evaluations in connection with projects. This class provides the linguistic means of expression for the evaluation of projects.	EvaluativesWissen

Table 24: Description of the subclasses of the class *EvaluativesWissen*

Figure 26 below shows the class *KonzeptuellesWissen* with the associated subclasses.

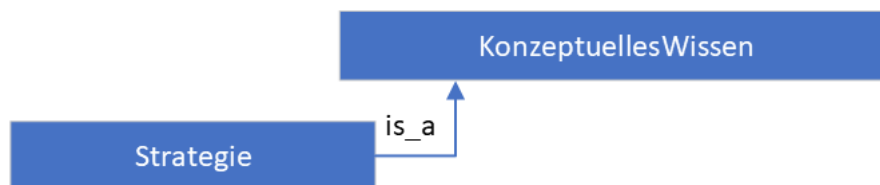


Figure 27: Subclasses of the class *KonzeptuellesWissen*

Table 25 below explains the subclass of the class *KonzeptuellesWissen*.

Class	Description	Subclass of
Strategie	The class <i>Strategie</i> is the superclass for all types of strategies.	KonzeptuellesWissen

Table 25: Description of the subclass of the class *KonzeptuellesWissen*

Figure 29 below shows the class *BeschreibungskonstituenteFürPRINCE2* with the associated subclasses.

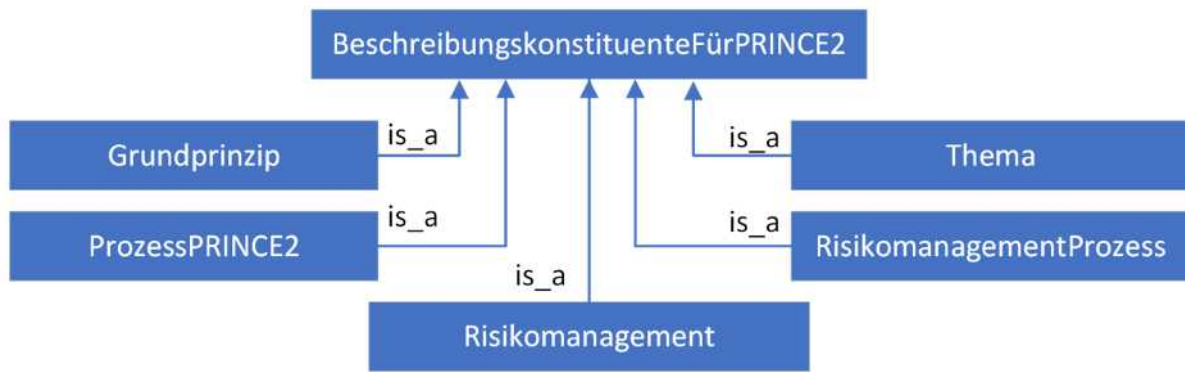


Figure 28: Subclasses of the class *BeschreibungskonstituenteFürPRINCE2*

Table 26 below explains the subclasses of the class *BeschreibungskonstituenteFürPRINCE2*.

Class	Description	Subclass of
Grundprinzip	The class <i>Grundprinzip</i> forms the superclass for PRINCE2's basic principles. These represent the essential guiding principles that must be observed during the implementation of a PRINCE2 project.	Beschreibungskonstituente FürPRINCE2
ProzessPRINCE2	The class <i>ProzessPRINCE2</i> is the superordinate class that represents PRINCE2's processes. These consist of a structured sequence of operations that are designed to achieve a defined goal.	Beschreibungskonstituente FürPRINCE2
Risikomanagement	The class <i>Risikomanagement</i> represents the superclass for all types of risk management. Note: As this is a polymorphic subsumption, two superordinate classes are indicated in the column on the right. Risk management comprises all measures to identify risks and to manage the processes associated with said risks in projects.	Beschreibungskonstituente FürPRINCE2 Methode
Risikomanagement Prozess	The class <i>RisikomanagementProzess</i> is the superclass for all types of PRINCE2 risk management processes. These processes represent a structured sequence of activities that include the identification, assessment, planning, and implementation of countermeasures and communication of risks.	Beschreibungskonstituente FürPRINCE2

Thema	The class <i>Thema</i> acts as a superclass for all PRINCE2 topics that are essential components of project management and must be dealt with continuously throughout the project lifecycle.	Beschreibungs konstituente Für PRINCE2
-------	--	--

Table 26: Description of the subclasses
of the class *BeschreibungskonstituenteFürPRINCE2*

Figure 29 below shows the class *BeschreibungskonstituenteFürScrum* with the associated subclasses.

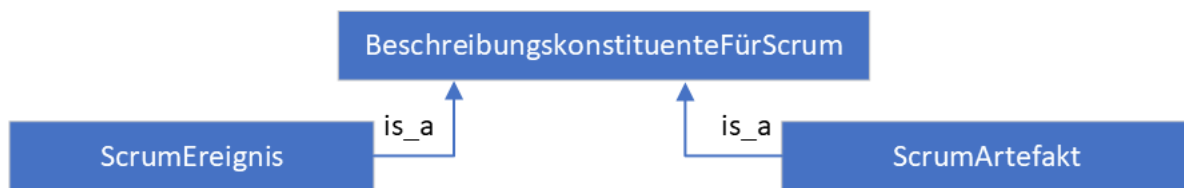


Figure 29: Subclasses of the class *BeschreibungskonstituenteFürScrum*

The class's division is based on the distinction between events and artefacts in the Scrum Guide by SCHWABER/SUTHERLAND (2020), pp. 7–12, which is considered standard literature for Scrum.

Class	Description	Subclass of
ScrumEreignis	The class <i>ScrumEreignis</i> represents the superclass for all Scrum events, such as Daily, Sprint, and Sprint Retrospective.	Beschreibungs konstituente FürScrum
ScrumArtefakt	The class <i>ScrumArtefakt</i> represents the superclass for all Scrum artefacts, such as the product increment, product backlog, and sprint backlog.	Beschreibungs konstituente FürScrum

Table 27: Description of the subclasses
of the class *BeschreibungskonstituenteFürScrum*

Figure 30 below shows the class *BeschreibungskonstituenteFürProdukte* with the associated subclasses.

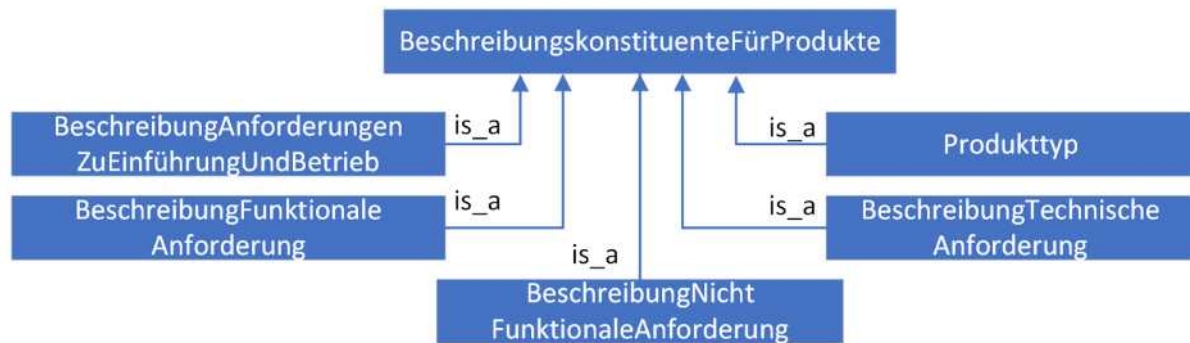


Figure 30: Subclasses of the class *BeschreibungskonstituenteFürProdukte*

We have based the subdivision of the class *BeschreibungskonstituenteFürProdukte* on the requirements laid out in the service descriptions on which this article rests. These service descriptions differentiate the requirements for the hybrid service bundle into the following areas:

- Requirements for implementation and operation
- Functional requirements
- Non-functional requirements
- Technical requirements

The subdivision of the requirements made by the service descriptions is also suitable for the further differentiation of the class *BeschreibungskonstituenteFürProdukte*. The following Table 28 assigns the requirement areas for the hybrid service bundle from the service descriptions to the respective class designations of the subclasses of the class *BeschreibungskonstituenteFürProdukte*.

Range of requirements	Class designation
Requirements for implementation and operation	<i>BeschreibungAnforderungenZuEinführungUndBetrieb</i>
Functional requirement	<i>BeschreibungFunktionaleAnforderung</i>
Non-functional requirement	<i>BeschreibungNichtFunktionaleAnforderung</i>
Technical requirement	<i>BeschreibungTechnischeAnforderung</i>

Table 28: Class designations of the requirement area

The advantage of this differentiation is that it creates a practical, safety-critical IT project ontology, as it is based on the underlying service descriptions.

In addition to the aforementioned differentiations for the class *BeschreibungskonstituenteFürProdukte*, the subclass *Produkttyp* is defined by the underlying PRINCE2 and risk management ontology. The class *Produkttyp* has been subordinated to both the class *BeschreibungskonstituenteFürPRINCE2* and the class *BeschreibungskonstituenteFürProdukte* through polymorphic subsumption. This polymorphic subsumption is based on the fact that the class *Produkttyp* is used semantically in the same way both as a description constituent of products in general and as a description constituent of the PRINCE2 project management method in particular. In PRINCE2, the product term is of central importance in order to differentiate between specialist products (which describe the specific deliverable, such as a safety-critical IT system) and management products (which are relevant for the creation of specialist products). Table 29 below explains the subclasses of the class *BeschreibungskonstituenteFürProdukte*.

Class	Description	Subclass of
Beschreibung Anforderungen ZuEinführung UndBetrieb	The class <i>BeschreibungAnforderungenZuEinführung UndBetrieb</i> is the superclass for requirements for introduction and operation.	Beschreibungs konstituente FürProdukte
Beschreibung Funktionale Anforderung	The class <i>BeschreibungFunktionaleAnforderung</i> is the superclass for functional requirements.	Beschreibungs konstituente FürProdukte
Beschreibung NichtFunktionale Anforderung	Die Klasse <i>BeschreibungNichtFunktionaleAnforderung</i> ist die superclass for non-functional requirements.	Beschreibungs konstituente FürProdukte
Beschreibung Technische Anforderung	The class <i>BeschreibungTechnischeAnforderung</i> is the superclass for technical requirements.	Beschreibungs konstituente FürProdukte
Produkttyp	The class <i>Produkttyp</i> is the superclass for all PRINCE2 product types. The class <i>Produkttyp</i> expresses the product concept of PRINCE2 and distinguishes between management and specialist products.	Beschreibungs konstituente FürProdukte Beschreibungs konstituente FürPRINCE2

Table 29: Description of the subclasses
of the class *BeschreibungskonstituenteFürProdukte*

At this point, we will explain polymorphic subsumption using Protégé via the example of the class *Produkttyp*. A polymorphic subsumption can be created in Protégé using the Class Expression Editor. The Class Expression Editor is called up in the class in which the polymorphic subsumption is to take place using the “SubClass Of” function and the “Plus” symbol, which is shown circled in red in Figure 31 below.

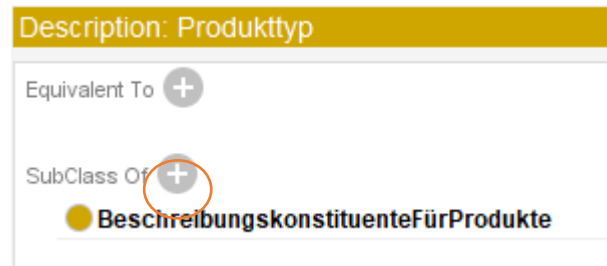


Figure 31: Calling up the function “SubClass Of” in Protégé

In the Class Expression Editor, the superordinate classes can be determined by entering the class names and using the logical operator “and”. After confirmation, the subclass is assigned to both classes as superordinate classes. Figure 32 below shows an example of this using the class *Produkttyp*.

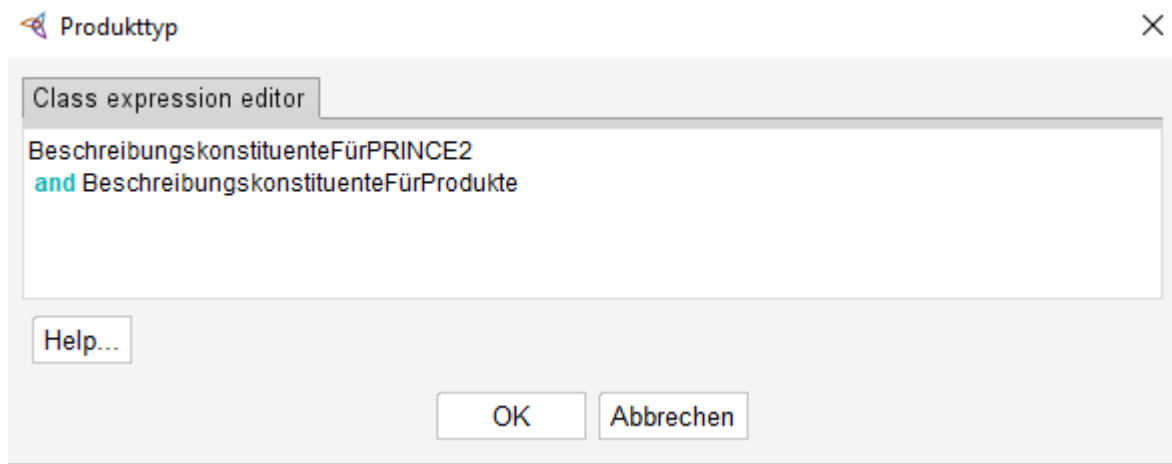


Figure 32: Class Expression Editor

Figure 33 below shows the class *BeschreibungskonstituenteFürProjekte* with the associated subclasses.

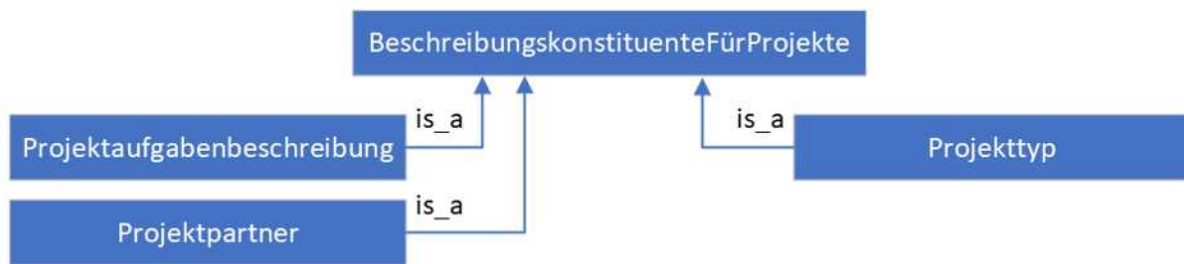


Figure 33: Subclasses of the class *BeschreibungskonstituenteFürProjekte*

Table 30 below explains the subclasses of the class *BeschreibungskonstituenteFürProjekte*.

Class	Description	Subclass of
Projektaufgabenbeschreibung	The class <i>Projektaufgabenbeschreibung</i> is the super-class for all types of project task descriptions.	BeschreibungskonstituenteFürProjekte
Projektpartner	The class <i>Projektpartner</i> is the superclass for all types of project partners. The project partners characterize a group of people who have a direct connection to a security-critical IT project.	BeschreibungskonstituenteFürProjekte Stakeholder
Projekttyp	The class <i>Projekttyp</i> is the superclass for all types of project types. A project type is used to characterize a project based on certain features.	BeschreibungskonstituenteFürProjekte

Table 30: Description of the subclasses of the class *BeschreibungskonstituenteFürProjekte*

Figure 34 below shows the class *BewertungskonstituenteFürProdukte* with the associated subclasses.

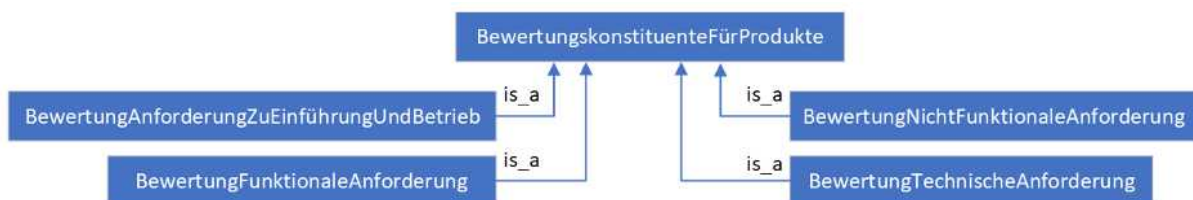


Figure 34: Subclasses of the class *BewertungskonstituenteFürProdukte*

The class *BewertungskonstituenteFürProdukte* provides the linguistic means of expression for the evaluation knowledge about a safety-critical IT project's requirements. The differentiation of this class is analogous to the differentiation of the class *BeschreibungskonstituenteFürProdukte*, which is based on the requirement areas of the underlying service descriptions. The class's aim is to represent the described requirements from the class *BeschreibungskonstituenteFürProdukte* in the class *BewertungskonstituenteFürProdukte* as evaluative knowledge.

The design decision to represent the requirements of a service description in the *BeschreibungskonstituenteFürProdukte* class as purely descriptive knowledge and to link them by means of non-taxonomic relations to the class *BewertungskonstituenteFürProdukte*, which represents the evaluating knowledge, allows requirements to be evaluated on the basis of the practical distinctions between the originally described requirement from the service description and the requirement actually implemented. This design decision enables the allocation of experience-based knowledge of a safety-critical IT project to specific requirements. Deviations between the description of a requirement and the evaluation of the fulfillment of a requirement are expressed by the subclasses of the class *SollIstAbweichungAnforderung*. Non-taxonomic relations are used to establish the link between description, evaluation, and target/actual deviation. The subclasses of the class *SollIstAbweichungAnforderung* are also differentiated into the requirement areas of requirements for implementation and operation, functional requirements, non-functional requirements, and technical requirements. Through this design decision, requirements demanded in the service description can be expressed with the service provided by the safety-critical IT project both as evaluation knowledge and as target/actual deviation, and through the safety-critical IT project ontology.

Figure 35 below summarizes the facts using the functional requirement of data maintenance as an example.

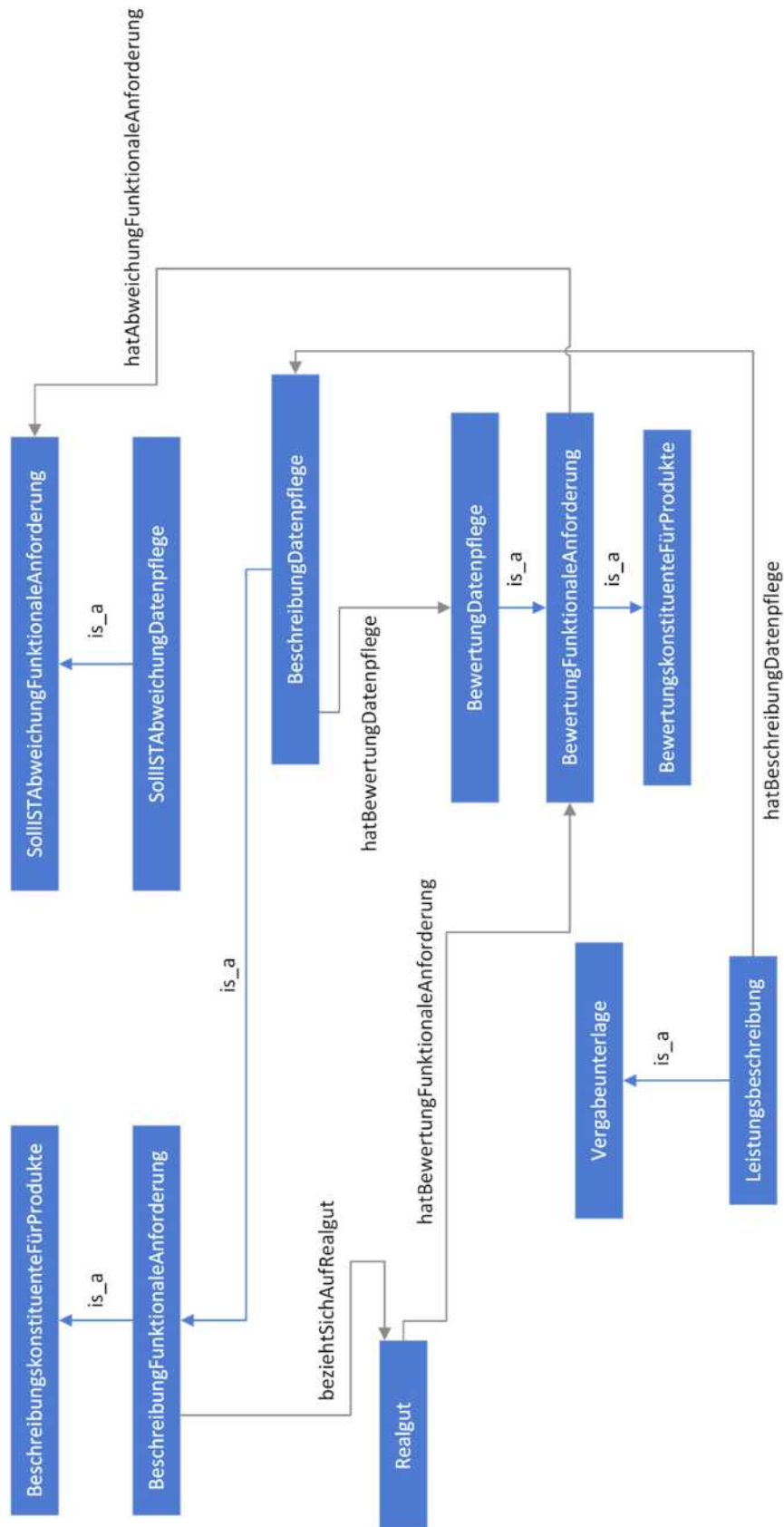


Figure 35: Illustration of the class construction using the example of the functional requirement of data maintenance

Figure 35 describes the following facts:

- The class *BeschreibungskonstituenteFürProdukte* forms the superclass for the class *BeschreibungFunktionaleAnforderung*, which in turn acts as the superclass for the class *BeschreibungDatenpflege*.
- The class *Leistungsbeschreibung*, which is a subclass of the class *Vergabeunterlage*, is linked to the class *BeschreibungDatenpflege* by means of a non-taxonomic relation *hatBeschreibungDatenpflege*. This construction is justified by the fact that the described requirements—such as data maintenance—originate from a service description, which in turn is part of the award documents.
- The class *BeschreibungFunktionaleAnforderung* is linked to the class *Realgut* by means of a non-taxonomic relation *beziehtSichAufRealgut*. The use of this non-taxonomic relation is intended to express the reference to the real goods required to fulfill the functional requirements.
- The class *Realgut* is linked by means of a non-taxonomic relation *hatBewertungFunktionaleAnforderung* with the class *BewertungFunktionaleAnforderung*, which represents the superclass for the class *BewertungDatenpflege*. This construction expresses the evaluation of the real assets used for the fulfillment of the functional requirements for data maintenance.
- The class *BewertungFunktionaleAnforderung* is linked to the class *SollIstAbweichungFunktionaleAnforderung* by means of a non-taxonomic relation called *hatAbweichungFunktionaleAnforderung*. This construction enables the representation of deviations between the original functional requirement and its actual fulfillment. The class *SollIstAbweichungFunktionaleAnforderung* is the superclass of the class *SollIstAbweichungDatenpflege*.
- The class *BeschreibungDatenpflege* is linked to the class *BewertungDatenpflege* by means of a non-taxonomic relation *hatBewertungDatenpflege*. This construction is intended to enable the evaluation of the fulfillment of the described functional requirements for data maintenance.

Table 31 below explains the subclasses of the class *BewertungskonstituenteFürProdukte*.

Class	Description	Subclass of
Bewertung Anforderung ZuEinführung UndBetrieb	The class <i>BewertungAnforderungZuEinführungUndBetrieb</i> is the superclass for the evaluations for the requirements for the introduction and operation of products.	Bewertungs konstituente FürProdukte
Bewertung Funktionale Anforderung	The class <i>BewertungFunktionaleAnforderung</i> is the superclass for all types of valuations for the functional requirements.	Bewertungs konstituente FürProdukte
Bewertung NichtFunktionale Anforderung	As a superclass, the class <i>BewertungNichtFunktionaleAnforderung</i> contains all types of evaluations for the non-functional requirements.	Bewertungs konstituente FürProdukte
Bewertung Technische Anforderung	The class <i>BewertungTechnischeAnforderung</i> is the superclass for all types of valuations for the technical requirements.	Bewertungs konstituente FürProdukte

Table 31: Description of the subclasses *BewertungskonstituenteFürProdukte*

Figure 36 below shows the class *BewertungskonstituenteFürProjekte* with the associated subclasses.

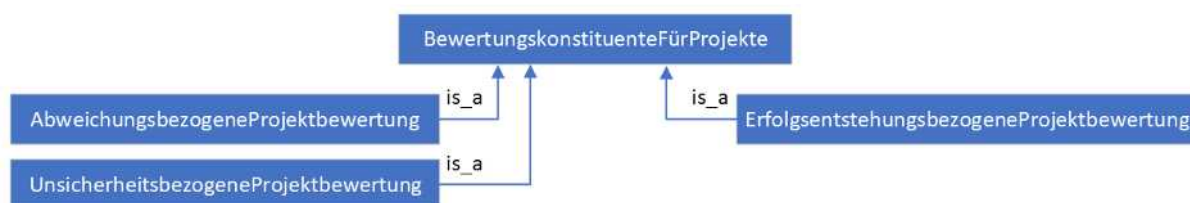


Figure 36: Subclasses of the class *BewertungskonstituenteFürProjekte*

The subclasses of the classes *AbweichungsbezogeneProjektbewertung*, *ErfolgseinstellungsbezogeneProjektbewertung* and *UnsicherheitsbezogeneProjektbewertung* provide linguistic means of expression for the following purposes:

- Risks of a safety-critical IT project (the class *Risikotyp* as a subclass of the class *UnsicherheitsbezogeneProjektbewertung*)

- Dependency relationships within a safety-critical IT project (the class *Abhängigkeitstyp* as a subclass of the class *UnsicherheitsbezogeneProjektbewertung*)
- Depicting relationship networks within a safety-critical IT project (the class *Beziehungsgeflecht* as a subclass of the class *UnsicherheitsbezogeneProjektbewertung*)
- Expressing success and failure factors within a safety-critical IT project (the classes *Erfolgsfaktor* and *Misserfolgsweg* as subclasses of the class *ErfolgsentstehungsbezogeneProjektbewertung*)
- Representing project quality variances in the form of target/actual variances (the class *SollIstAbweichungAnforderung* as a subclass of the class *AbweichungsbezogeneProjektbewertung*)

Table 32 below explains the subclasses of the class *BewertungskonstituenteFürProjekte*.

Class	Description	Subclass of
Abweichungsbezogene Projektbewertung	The class <i>AbweichungsbezogeneProjektbewertung</i> is the superclass for all types of deviation-related project evaluations.	Bewertungskonstituente FürProjekte
Unsicherheitsbezogene Projektbewertung	The class <i>UnsicherheitsbezogeneProjektbewertung</i> is the superclass for all types of uncertainty-related project evaluations.	Bewertungskonstituente FürProjekte
ErfolgsentstehungsbezogeneProjektbewertung	The class <i>ErfolgsentstehungsbezogeneProjektbewertung</i> is the superclass for all types of performance-related related project evaluations.	Bewertungskonstituente FürProjekte

Table 32: Description of the subclasses *BewertungskonstituenteFürProjekte*

Figure 37 below shows the class *ImmateriellesRealgut* with the associated subclasses.

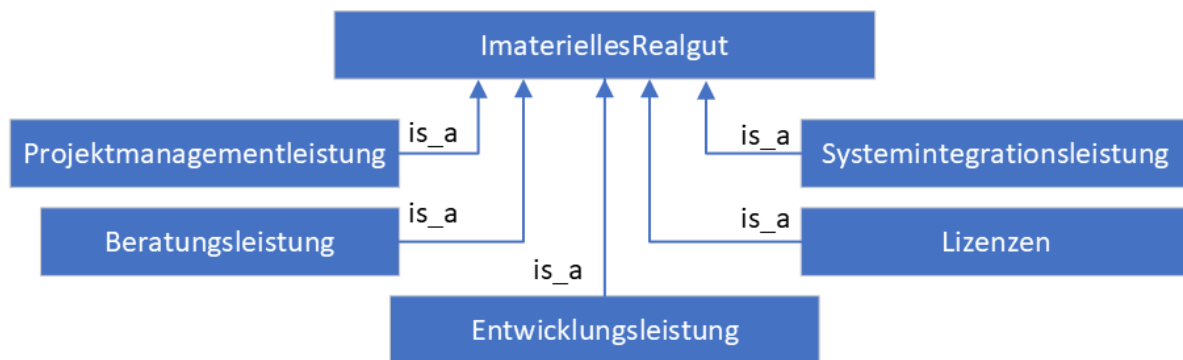


Figure 37: Subclasses of the class *ImmateriellesRealgut*

The differentiation of the class *ImmateriellesRealgut* into the other subclasses is essentially based on the performance specifications used. These underlying specifications require various intangible real assets (both in the sense of production factors and products), which are differentiated into the following areas:

- Project management services
- Consulting services
- Development services
- Licenses
- System integration services

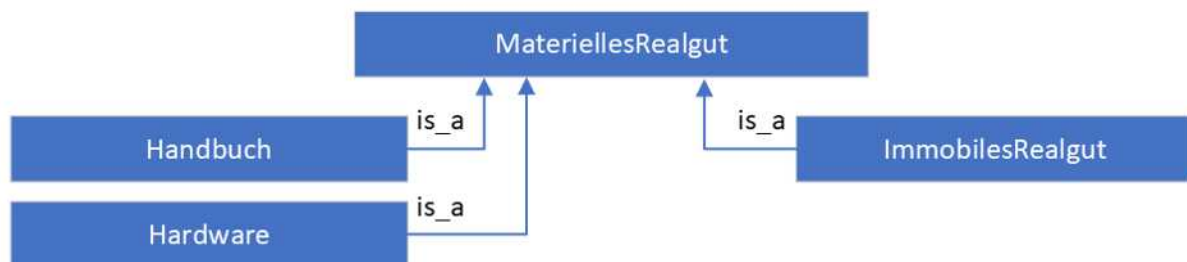
These service types represent the subclasses of the class *ImmateriellesRealgut* and are explained in Table 33 below.

Class	Description	Subclass of
Projektmanagementleistung	The class <i>Projektmanagementleistung</i> is the superclass for all types of project management services.	Immaterielles Realgut
Beratungsleistung	The class <i>Beratungsleistung</i> is the superclass for all types of consulting services.	Immaterielles Realgut
Entwicklungleistung	As a superclass, the class <i>Entwicklungsleistung</i> contains all types of development services.	Immaterielles Realgut

Lizenzen	The class <i>Lizenzen</i> is the superclass for all types of licenses.	Immaterielles Realgut
Systemintegrationsleistung	The class <i>Systemintegrationsleistung</i> is the superclass for all types of system integration services.	Immaterielles Realgut

Table 33: Description of the subclasses of the class *ImmateriellesRealgut*

Figure 38 below shows the class *MateriellesRealgut* with the associated subclasses.

Figure 38: Subclasses of the class *MateriellesRealgut*

Analogous to the differentiation of the class *ImmateriellesRealgut*, the subdivision of the class *MateriellesRealgut* is primarily based on the various tangible real assets required in the specifications, subdivided into the following areas:

- Manual
- Hardware
- Immovable real assets

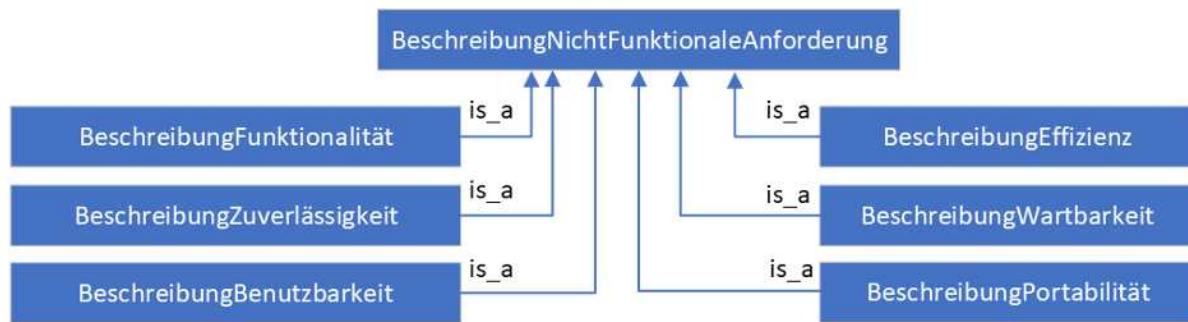
These areas represent the subclasses of the class *MateriellesRealgut* and are explained in Table 34 below.

Class	Description	Subclass of
Handbuch	The class <i>Handbuch</i> is the upper class for all types of manuals.	Materielles Realgut
Hardware	The class <i>Hardware</i> is the upper class for all types of hardware.	Materielles Realgut

ImmobilienRealgut	The class <i>ImmobilienRealgut</i> is the superclass of all types of immovable real assets.	Materielles Realgut
-------------------	---	---------------------

Table 34: Description of the subclasses of the class *MateriellesRealgut*

Figure 39 below shows the class *BeschreibungNichtFunktionaleAnforderung* with the associated subclasses.

Figure 39: Subclasses of the class *BeschreibungNichtFunktionaleAnforderung*

Non-functional requirements are divided into different categories of characteristics in accordance with ISO standard 9126. The underlying performance descriptions are based on this standard; cf. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2000), pp. 7–13. This justifies the differentiation of the subclasses of the class *BeschreibungNichtFunktionaleAnforderung* into the characteristics explained in Table 35 below.

Characteristic	Definition
Functionality	The functionality characteristic describes that the expected functionality and the intended benefit are fulfilled.
Reliability	The reliability characteristic specifies the product's ability to operate without faults and to maintain the specified level of performance under standard operating conditions.
Usability	The usability characteristic specifies the effort required to learn how to use the product.

Efficiency	The characteristic efficiency describes the product's ability to achieve an appropriate performance, taking into account resources such as time, storage space, or energy consumption, and its capacity to meet the time behavior requirements
Maintainability	The maintainability characteristic specifies the product's ability to be changeable in order to be able to make corrections, improvements, and modifications.
Portability	The portability feature specifies the product's ability to be transferred from one environment to another.

Table 35: Features of the ISO standard 9126

The following Table 36 explains the subclasses of the class based on the characteristics defined in Table 35, *BeschreibungNichtFunktionaleAnforderung*.

Class	Description	Subclass of
Beschreibung Funktionalität	The class <i>BeschreibungFunktionalität</i> is the superclass for all types of descriptions of non-functional requirements that relate to the functionality feature area.	Beschreibung NichtFunktionale Anforderung
Beschreibung Zuverlässigkeit	The class <i>BeschreibungZuverlässigkeit</i> is the superclass for all types of descriptions of non-functional requirements that relate to the reliability feature area.	Beschreibung NichtFunktionale Anforderung
Beschreibung Benutzbarkeit	The class <i>BeschreibungBenutzbarkeit</i> is the superclass for all types of descriptions of non-functional requirements that relate to the usability feature area.	Beschreibung NichtFunktionale Anforderung
Beschreibung Effizienz	The class <i>BeschreibungEffizienz</i> is the superclass for all types of descriptions of non-functional requirements that relate to the efficiency feature area.	Beschreibung NichtFunktionale Anforderung

Beschreibung Wartbarkeit	The class <i>BeschreibungWartbarkeit</i> is the super-class for all types of descriptions of non-functional requirements that relate to the maintainability feature area.	Beschreibung NichtFunktionale Anforderung
Beschreibung Portabilität	The class <i>BeschreibungPortabilität</i> is the super-class for all types of descriptions of non-functional requirements that relate to the portability feature area.	Beschreibung NichtFunktionale Anforderung

Table 36: Description of the subclasses
of the class *BeschreibungNichtFunktionaleAnforderung*

The risk type class serves as a linguistic means of expression for describing risks in safety-critical IT projects. It is differentiated into the subclasses *MittelbaresRisiko*, *UnmittelbaresRisiko*, and *Unsicherheit*. This distinction results from the fact that indirect risks arise from external influences and can be influenced only to a limited extent, while direct risks have internal project causes and can generally be influenced more strongly. Uncertainties can be identified but not yet assessed.

Figure 40 below shows an example of the taxonomic and non-taxonomic relations of the subclasses of the class *Risikotyp* in the area of functional requirements. This makes it clear which conceptual decisions underlie the safety-critical IT project ontology in order to linguistically structure the risks that can arise for each requirement.

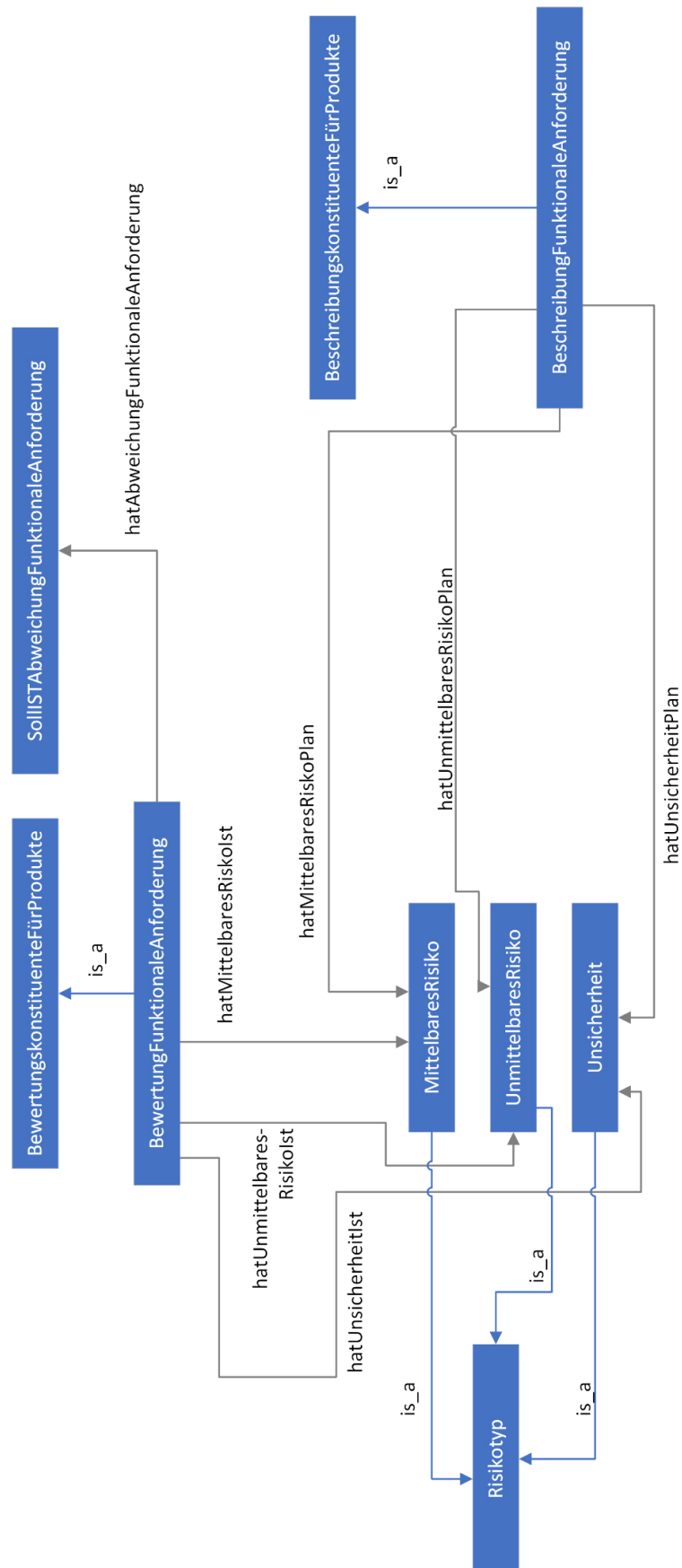


Figure 40: Structuring risks using the example of functional requirements

Figure 40 shows the following facts:

- The class *BeschreibungFunktionaleAnforderung* is linked to the class *MittelbaresRisiko* with a non-taxonomic relation (*hatMittelbaresRisikoPlan*). This construction makes it possible to assign the described functional requirements to the planned indirect risks.
- The class *BeschreibungFunktionaleAnforderung* is linked to the class *UnmittelbaresRisiko* with a non-taxonomic relation (*hatUnmittelbaresRisikoPlan*). This construction makes it possible to assign the described functional requirements to the planned immediate risks.
- The class *BeschreibungFunktionaleAnforderung* is linked to the class *Unsicherheit* with a non-taxonomic relation (*hatUnmittelbareUnsicherheitPlan*). This construction makes it possible to assign the described functional requirements to the planned uncertainties.
- The class *BewertungFunktionaleAnforderung* is linked to the class *mittelbaresRisiko* by a non-taxonomic relation (*hatmittelbaresRisikoIst*). This construction makes it possible to allocate the functional requirements to the indirect risks that have actually occurred.
- The class *BewertungFunktionaleAnforderung* is linked to the class *UnmittelbaresRisiko* by a non-taxonomic relation (*hatUnmittelbaresRisikoIst*). This construction makes it possible to assign the functional requirements to the direct risks that have actually occurred.
- The class *BewertungFunktionaleAnforderung* is linked to the class *Unsicherheit* by a non-taxonomic relation (*hatUnsicherheitIst*). This construction makes it possible to assign the functional requirements to the uncertainties that have actually occurred.

Table 37 below explains the subclasses of the class *Risikotyp*:

Class	Description	Subclass of
UnmittelbaresRisiko	The class <i>UnmittelbaresRisiko</i> is the superclass for all types of immediate risks.	Risikotyp
MittelbaresRisiko	The class <i>MittelbaresRisiko</i> is upper class for all types of indirect risks.	Risikotyp

Unsicherheit	The class <i>Uncertainty</i> is the superclass for all types of uncertainties	Risikotyp
--------------	---	-----------

Table 37: Description of the subclasses of the class *Risikotyp*

3.2.3.5 Construction of non-taxonomic relations

Below, we discuss the construction of properties, which requires a distinction to be made between non-taxonomic relations and attributes. We will initially only consider the construction of non-taxonomic relations (also referred to as “relations” for short); we explain the construction of attributes later, in chapter 3.2.3.6.

In the following explanations, we will give the non-taxonomic relations a tabular representation structured as follows:

Relation name	Domain	Range

Table 38: Table structure for the explanation of non-taxonomic relations

A non-taxonomic relation has a relation designation and is specified with the associated domain (pre-range) and range (post-range). Figure 41 below shows said relation’s relevant components.

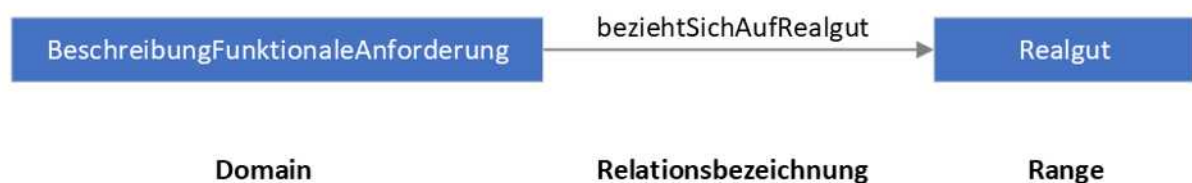


Figure 41: Components of a non-taxonomic relation

Figure 41 shows an example of the non-taxonomic relation *refersToRealGood*, which has the class *BeschreibungFunktionaleAnforderung* assigned as the domain and the class *Realgut* as the range. Using the logical expressions AND and OR, further classes can be added to both the domain and the range. In the above example, the relation *beziehtSichAufRealgut* connects individuals of the class *BeschreibungFunktionaleAnforderung* (domain) with individuals of the class *Realgut* (range).

Some non-taxonomic relations are already predefined in this PM domain ontology and for the PRINCE2 and risk management ontology. Yet the specific requirements of safety-critical IT projects make these insufficient. Examples of a specific requirement for the safety-critical IT project ontology are the necessary linguistic means of expression used in the underlying service descriptions. By introducing additional classes in the safety-critical IT project ontology, non-taxonomic relations must also be constructed in order to increase the safety-critical IT project ontology's expressiveness.

In the following explanations, we deal only with the newly constructed non-taxonomic relations; we will not discuss the predefined relations further.

Below, we explain the construction steps required to create a non-taxonomic relation in Protégé. A non-taxonomic relation is created under “Object Properties”, as shown in Figure 42.

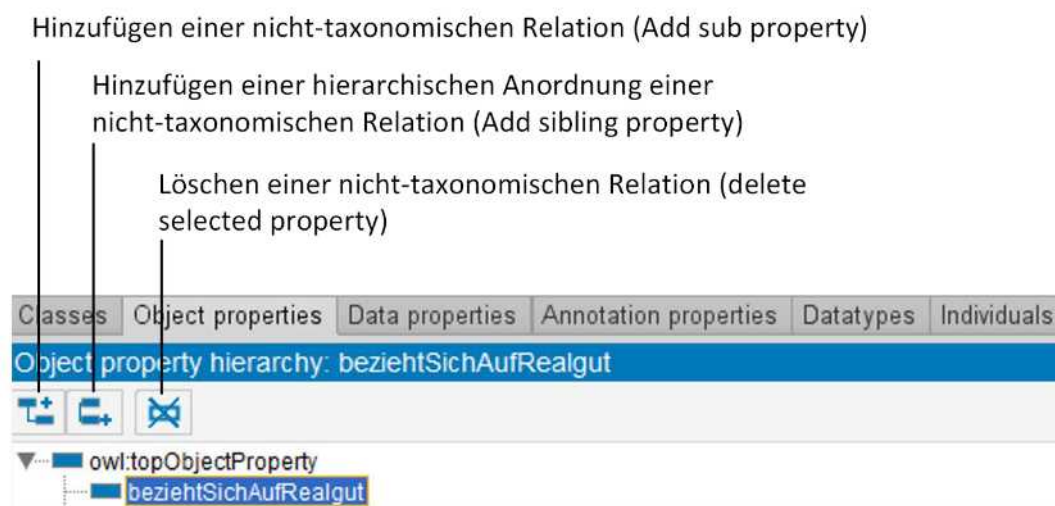


Figure 42: Object properties in Protégé

Protégé offers two ways to construct non-taxonomic relations. In this article, we use the “Add Sub Property” option is used to construct a non-taxonomic relation. However, it is also possible to construct a non-taxonomic relation that stands in a subsumption relationship to another non-taxonomic relation. When constructed as a subsumption relationship (this relates to the “Add sibling property” option in Figure 42), the subordinate non-taxonomic relation inherits the property of the superordinate non-taxonomic relation; cf. DEBELLIS (2021), p. 22. In the safety-critical IT project ontology, no non-taxonomic relation is constructed that has a subsumption relationship. This design decision is justified by the fact that it has not proved expedient to map a subsumption relationship when specifying non-taxonomic relations. Although there are non-taxonomic relations for which the specification of a subsumption ratio would have been appropriate, the use such a ratio is irrelevant in a CBR system.

As soon as the non-taxonomic relations have been constructed, all non-taxonomic relations are subordinated to the relation *owl:topObjectProperty* specified by Protégé as standard, as shown in Figure 42 as an example for the non-taxonomic relation *bezieht SichAufRealgut*.

Figure 43 below shows the construction of the classes for the domain of the non-taxonomic relation *beziehtSichAufRealgut* using the Class Expression Editor. The Class Expression Editor can be used to assign the class for the “range” area in a similar way to the assignment of the domain.

In the above example, several classes are assigned to this non-taxonomic relation in the domain, which are connected with the logical expression OR. This means that each of the named classes is permitted, making the union of their individuals available for instantiation in the relation’s domain.

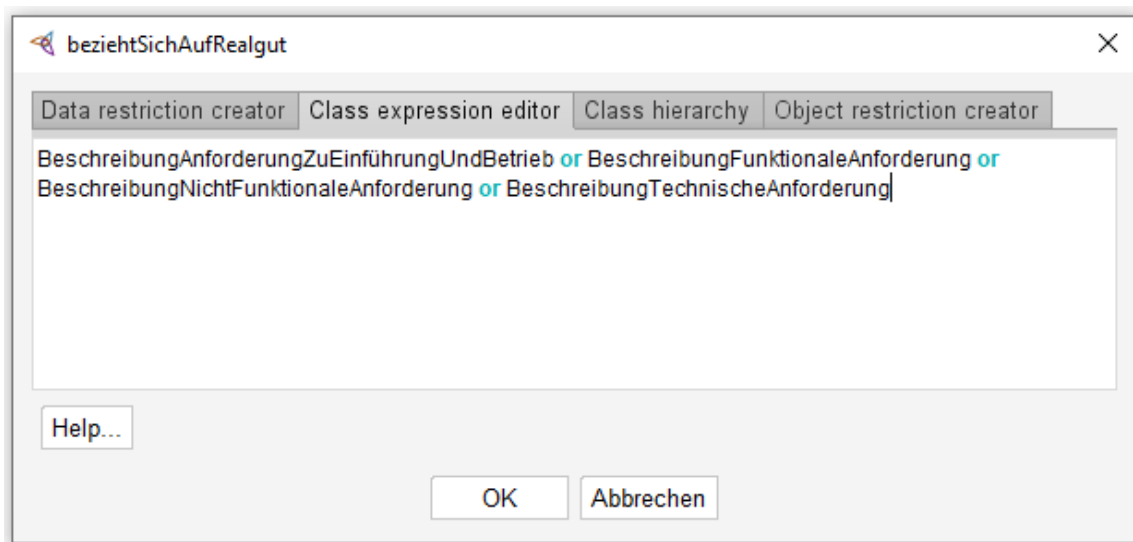


Figure 43: Class Expression Editor
with the assignment of a domain to a non-taxonomic relation

The result of the assignment can be viewed in the “Description: *beziehtSichAufRealgut*” area, as illustrated in Figure 44 below.

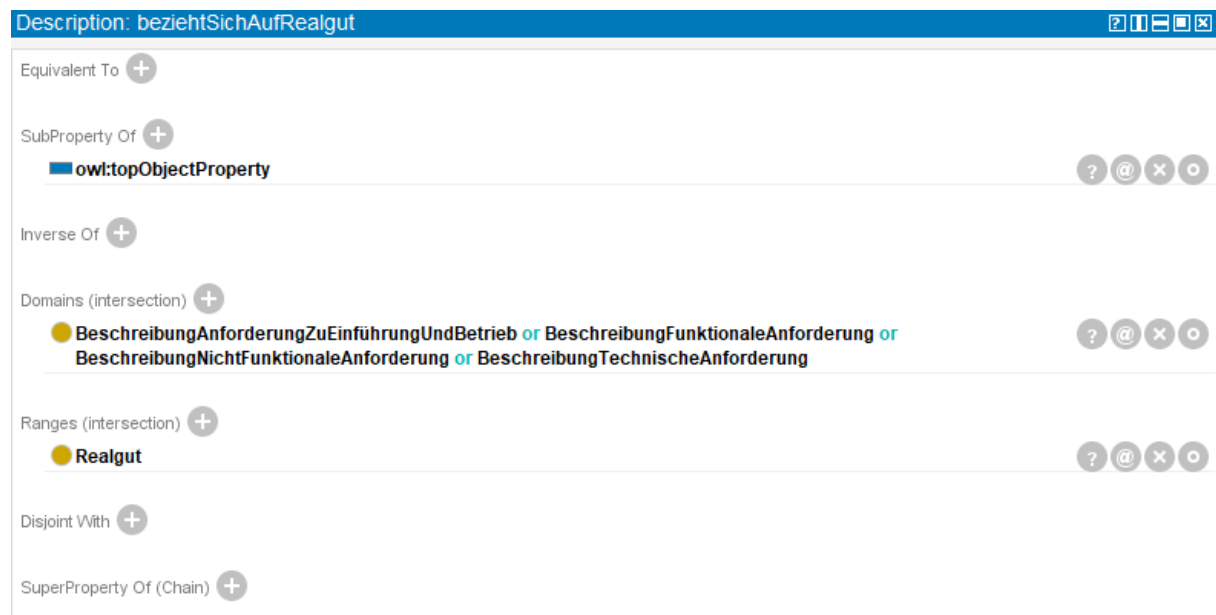


Figure 44: Description of a non-taxonomic relation in Protégé

The expressive power of non-taxonomic relations can be extended by further characteristics in Protégé. Protégé offers the following characteristics for non-taxonomic relations: functional, inverse functional, transitive, symmetric, asymmetric, reflexive, and irreflexive. These are selected in a separate selection window, which is labeled “Characteristics”. We did not use these characteristics for the construction of the safety-critical IT project ontology. Similar to the use of the subsumption of non-taxonomic relations, there are non-taxonomic relations for which the use of the aforementioned characteristics might have been appropriate, but which are irrelevant in a CBR system. The use of characteristics is also discussed controversially in some sources. For example, DEBELLIS (2021), p. 26, recommends the considered use of the characteristic “reflexive”, due to the effect on the reasoning component (reasoner). For a more detailed explanation of the characteristics, please refer to DEBELLIS (2021), pp. 24–26.

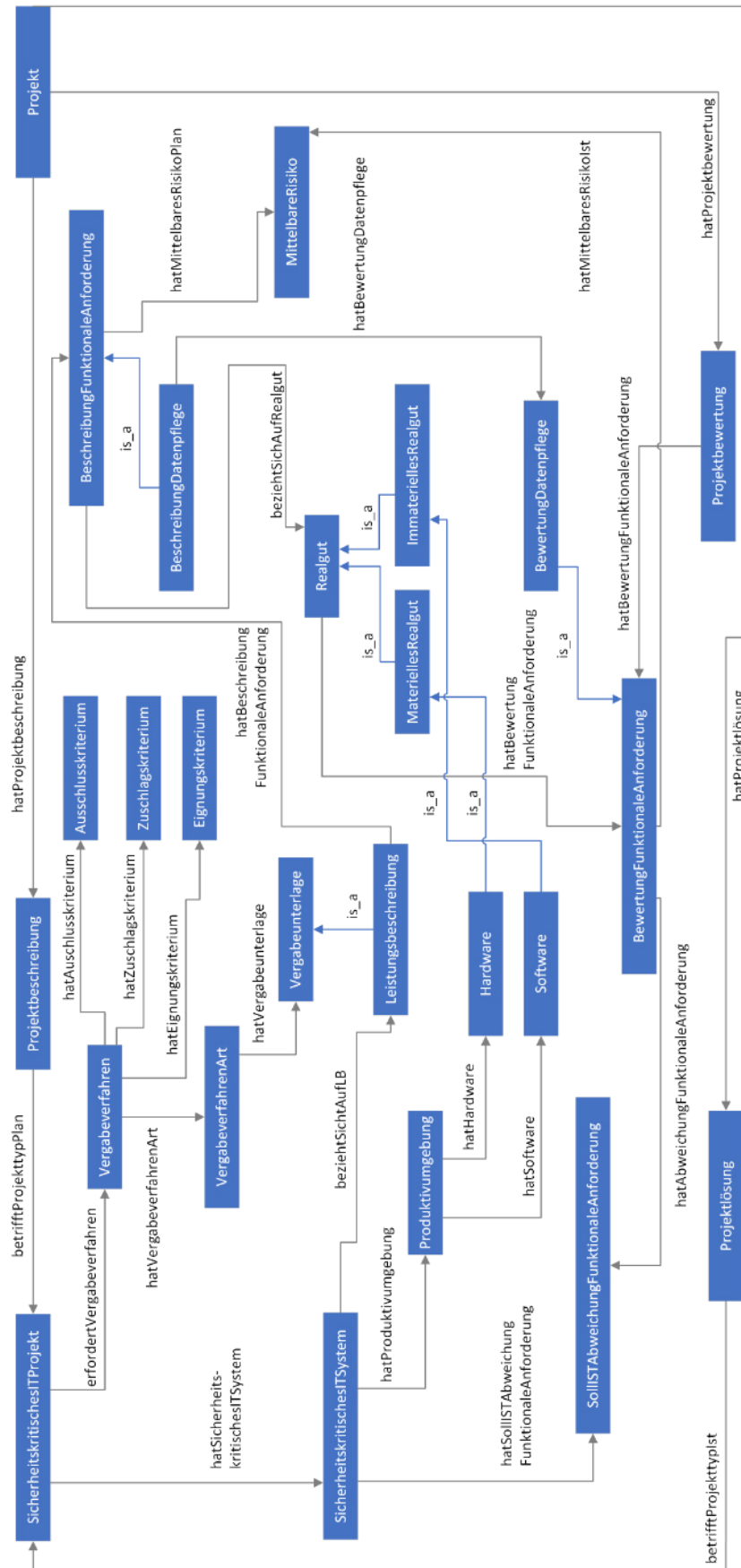


Figure 45: Exemplary excerpt from the safety-critical IT project ontology

Figure 45 shows the following facts with regard to non-taxonomic relations:

- The class *Projektbeschreibung* and the class *SicherheitskritischesITProjekt* are linked by means of a non-taxonomic relation (*betrifftProjekttypPlan*). The class *Projektbeschreibung* is a central class for the CBR system, and the construction of the non-taxonomic relation (*betrifftProjekttypPlan*) provides the linguistic means of expression for describing the planned project characteristics on the basis of the Project Type class and assigning them to the Project Description.
- The class *SicherheitskritischesITProjekt* and the class *Vergabeverfahren* are linked by means of a non-taxonomic relation (*erfordertVergabeverfahren*). This design decision provides linguistic means of expression for assigning a safety-critical IT project to an award procedure.
- The class *Vergabeverfahren* is linked to the classes *Ausschlusskriterium*, *Zuschlagskriterium*, and *Eignungskriterium* by the non-taxonomic relations *hatAusschlusskriterium*, *hatZuschlagskriterium*, and *hatEignungskriterium*, respectively. This construction results from the fact that an award procedure has exclusion criteria, award criteria, and suitability criteria for the evaluation of potential bidders.
- The class *Vergabeverfahren* is linked to the class *VergabeverfahrenArt* by means of a non-taxonomic relation, namely *hatVergabeverfahrenArt*. This construction makes it possible to assign a specific award procedure type to an award procedure.
- The class *VergabeverfahrensArt* is linked via a non-taxonomic relation *hatVergabeunterlage* to the class *Vergabeunterlage*. This construction makes it possible to express the documents provided in the context of a specific award procedure, such as a service description (expressed as a class *Leistungsbeschreibung*) in the safety-critical IT project ontology.
- The class *SicherheitskritischesITProjekt* is linked via a non-taxonomic relation *hatSicherheitskritischesITSystem* to the class *SicherheitskritischesITSystem*. This design decision is in line with the definition of a safety-critical IT project presented in this article, according to which a safety-critical IT project has a safety-critical IT system as its deliverable.
- The class *SicherheitskritischesITSystem* is linked via a non-taxonomic relation *beziehtSichAufLB* to the class *Leistungsbeschreibung*. This design decision is intended to express the performance description to which the safety-critical IT system refers.

- The class *Leistungsbeschreibung* is linked via a non-taxonomic relation *hatBeschreibungFunktionaleAnforderung* to the class *BeschreibungFunktionaleAnforderung*. The aim of this construction is to reference the specific functional requirements formulated in a service description. In the context of public procurement procedures, requirements are formulated in performance specifications, as already explained in this article, which is why this design decision serves to express this fact for the area of functional requirements.
- The class *SicherheitskritischesITSystem* is connected to the class *Produktivumgebung* by the non-taxonomic relation *hatProduktivumgebung*. The aim of this construction is to express the productive environment of a safety-critical IT system. The class *Produktivumgebung* is connected to the classes *Hardware* and *Software* by the two non-taxonomic relations *hatHardware* and *hatSoftware*. Both classes are subclasses of the class *Realgut*. This is intended to express which tangible and intangible real assets, in this case software and hardware, make up the productive environment of a safety-critical IT system
- The class *Projektlösung* is linked to the class *SicherheitskritischesITProjekt* by means of the non-taxonomic relation *betrifftProjekttypIst*. This construction makes it possible to access all relevant classes and properties in order to express the actual project solution of a safety-critical IT project. In addition, the non-taxonomic relation *hatAbweichungFunktionaleAnforderung*, which links the class *BewertungFunktionaleAnforderung* with the class *SollISTAbweichungFunktionaleAbweichung*, provides the means to express the deviation from the planned solution in addition to the realized project solution.

A complete list of all non-taxonomic relations of the safety-critical IT project ontology can be found in SETHUPATHY (2024), pp. 243–257.

It should be noted that computer-aided knowledge management systems generally do not support N -digit relations with $N > 2$; only two-digit non-taxonomic relations are common. This restriction means that, for example, it is not possible to specify a non-taxonomic relation that states that a specific safety-critical IT system X requires software Y that only works with hardware Z (ternary relation). Representing this situation requires two non-taxonomic relations, whereby the class *Software* occurs in the domain or range of the two non-taxonomic relations (*hatSoftware*, *benötigtHardware*).

3.2.3.6 Constructing attributes

We describe the construction of the attributes below and show them in a table, which is structured as follows:

Domain	Attribute name	Range

Table 39: Table structure for the explanation of the attributes

An attribute has an attribute name and is defined with its domain (pre-range) and its range (post-range). In contrast to non-taxonomic relations, the range does not refer to a class, but to a data type. A data type defines how many bytes a variable occupies from its address in a computer system's main memory and how the bit pattern of these bytes is interpreted. We use only primitive data types in this article, as these are both supported by the CBR tool jCORa and recommended in the relevant technical literature. We decided to restrict ourselves to primitive data types in order to ensure greater compatibility with other systems and better interchangeability of the data.

Figure 46 below shows the relevant components of an attribute in an exemplary manner using the attributes of the class *SicherheitskritischesITProjekt*.

SicherheitskritischesITProjekt: Klasse
hatProjektName: String benötigtSicherheitsüberprüfung: Boolean hatProjektAuftraggeber: String hatTCVBC: Integer hatCPVCode: String hatProjektAnwalt: Boolean

Figure 46: Exemplary representation of the attributes of the class *SicherheitskritischesITProjekt*

Some attributes are already pre-constructed in the underlying PM domain ontology and the PRINCE2 and risk management ontology. However, the existing attributes are not sufficient for the safety-critical IT project ontology, partly due to the additionally constructed classes.

The attributes are constructed in Protégé in the “Data Properties” area, as shown in Figure 47 below.

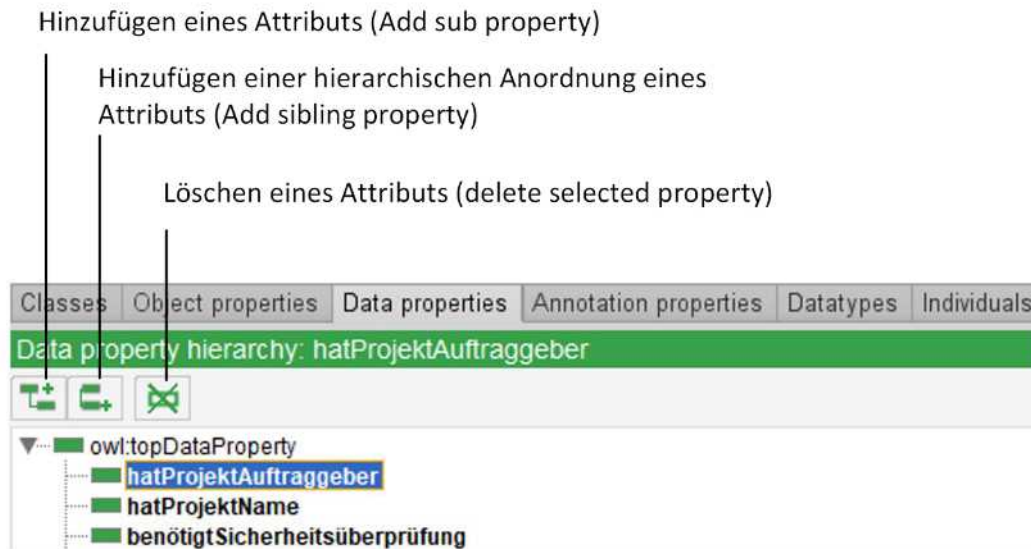


Figure 47: Data properties in Protégé

Protégé offers two ways to construct attributes. For this article, we used the “Add sub property” option. However, it is also possible to construct attributes that are in a subsumption relationship with other attributes, by using the “Add sibling property” option.

Once the attributes have been constructed, they are all subordinated to the *owl:topDataProperty* attribute specified by Protégé by default, as shown in Figure 47 as an example for the *hatProjektName*, *benötigtSicherheitsüberprüfung*, and *hatProjektAuftraggeber* attributes.

Figure 48 below uses the Class Expression Editor to illustrate the construction of the attribute *benötigtSicherheitsüberprüfung*, to which the class *SicherheitskritischesITProjekt* is assigned as the domain. The Boolean data type is assigned in the range of the attribute, as shown in Figure 49.

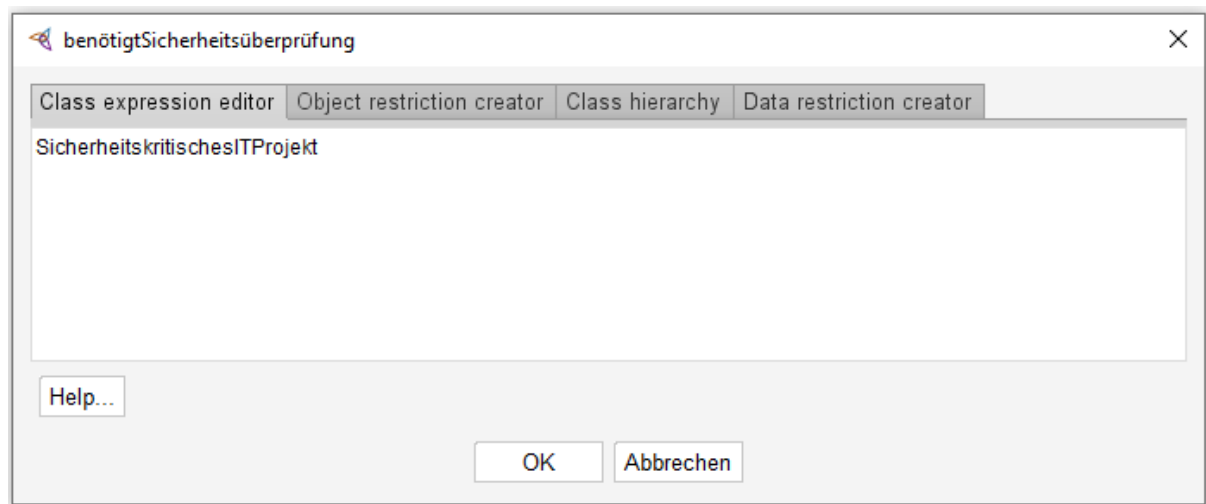


Figure 48: Class Expression Editor for assigning a domain to an attribute

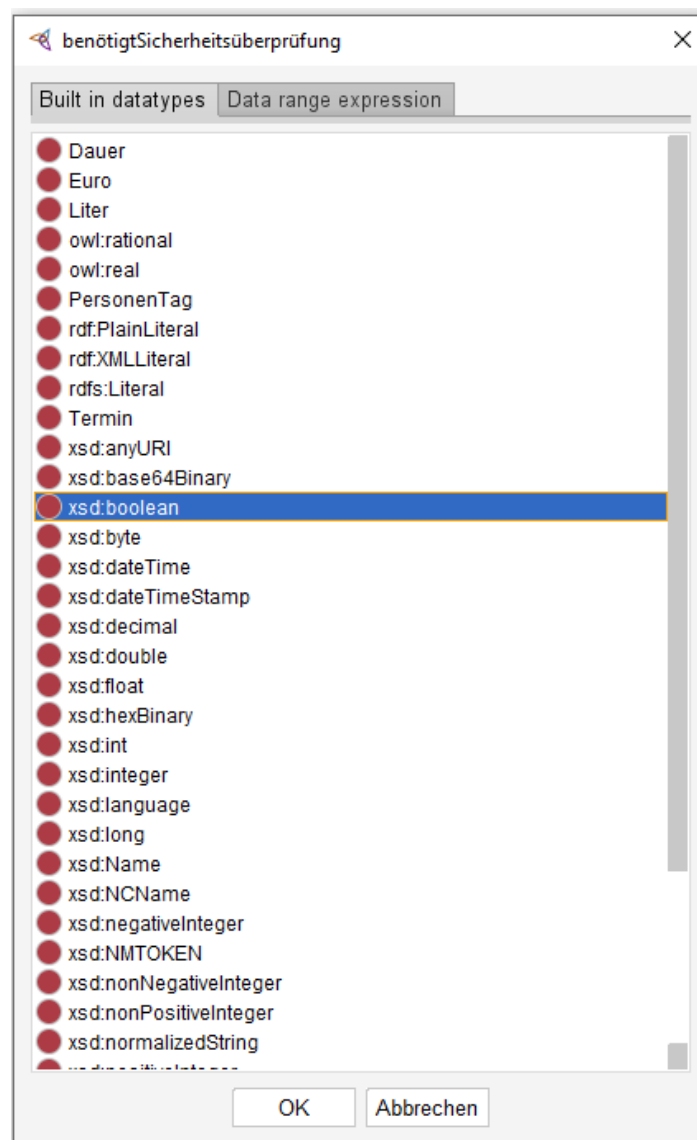


Figure 49: Assignment of a data type in the post area of the attribute

Protégé supports the following primitive data types for mapping:

Data type	Explanation
String	The “String” data type is used when an attribute takes a character string as its value.
Float	The “Float” data type is used when an attribute takes a floating point number as its value.
Integer	The “Integer” data type is used when an attribute takes an integer as its value.
Boolean	The “Boolean” data type is used when an attribute assumes a truth value (true, false).

Table 40: Explanation of the primitive data types

Protégé also offers the option of creating one’s own data types, with the underlying PM domain ontology offering individual data types such as duration, euro, and liter. However, we have created no such individual data types for this safety-critical IT project ontology. One reason for this is that the CBR tool jCORA does not support any additionally created data types without adjustments to the program’s source code. Moreover, most attributes can be mapped with the primitive data types. This view is also followed in DEBELLIS (2021), p. 48, where the use of primitive data types is the rule. Should it nevertheless be necessary to create an independent data type, the construction of a new class is recommended. Exemplary design decisions for the attributes are justified on the basis of the class *Vergabeverfahren*. Figure 50 below shows the class *Vergabeverfahren* with its attributes and the assigned data types.

Vergabeverfahren: Klasse
hatCPVCode: String
wurdeBereitsAufgehoben: Boolean
hatLinkZuBekanntmachung: String
hatVorgeschaltetenTeilnahmeantrag: Boolean
hatErfüllungsOrt: String
hatArtDesAuftrags: String
hatLose: Boolean
hatGeschätztenWert: Integer
hatBezeichnungDesAuftrags: String
hatLaufzeitDerAngefragtenLeistung: Integer
hatNUTSCode: String
hatArtDesÖffentlichenAuftraggeber: String
hatHaupttätigkeiten: String
überschreitetSchwellenwert: Boolean
hatVergabeanwalt: Boolean
istKomplexesVergabeverfahren: Boolean
istBundesausschreibung: Boolean
hatArtDesAuftraggebers: String
hatTCVPlan: Integer
betrifftBundesland: Boolean

Figure 50: The class *Vergabeverfahren* with the assigned attributes

We have based the construction of the attributes for the class *Vergabeverfahren* on the public announcements of public tenders, and have extended them by additional attributes. We explain the design decisions of these attributes below.

- The attribute *hatCPVCode* expresses the tender's unique identification number. Although the CPV code is a combination of numbers, its last number is separated by the “-” character. In addition, an integer assignment does not take the leading zeros into account; as this could falsify the CPV code, we have selected the string data type for this assignment.
- The attribute *wurdeBereitsAufgehoben* expresses whether the award procedure was already put out to tender in the past and canceled while in process. Its data type is Boolean, as the specification is sufficient as a truth value.
- The attribute *hatLinkZuBekanntmachung* represents the link to an Internet address. Calling this up in a web browser allows the information on the award procedure's public notice to be viewed. It is a string data type, as the link represents a character string that refers to a website.

- The attribute *hatVorgeschaltetenTeilnahmeantrag* expresses the property that a participation request was submitted before the award procedure. Either a previous request to participate exists, or it does not—this makes its data type Boolean, as the truth value suffices to represent the property.
- The attribute *hatErfüllungsOrt* specifies the location where the award procedure's service is to be provided. It is a string data type, as the location designation is a character string.
- The attribute *hatArtDesAuftrag* specifies the classification of the award procedure's requested service. It is a string data type, as the classification is a character string, such as "services".
- The *hatLose* attribute indicates whether the award procedure has been divided into several "lots". Its data type is Boolean, as a truth value is sufficient to indicate whether the award procedure has lots.
- The attribute *hatGeschätztenWert* specifies the award procedure's estimated value from the perspective of the contracting authority. It is an integer value data type, as the specification of the estimated value is not a floating point number, but an estimated integer.
- The attribute *hatBezeichnungDesAuftrags* specifies the award procedure's name by using a character string; we have therefore assigned it to the string data type.
- The attribute *hatLaufzeitDerAnfragragtenLeistung* specifies the duration of the award procedure's requested service. It is an integer data type, because the duration's unit of measurement—the number of years—is expressed as an integer.
- The attribute *hatNUTSCode* expresses the code "Nomenclature des unités territoriales statistiques", which is a composite key consisting of digits and letters; cf. EUROSTAT (2022), p. 4. The purpose of the NUTS code is to enable a clear division of the geographical areas of official statistics in the European Union. It is mandatory for alerts in the European Union. This attribute is a string data type, as it is a combination of numbers and letters. A list of the current NUTS codes for Germany can be found in EUROSTAT (2022), pp. 30–40. The NUTS code can be used to establish a clear geographical reference to a tender in the European Union and to search for specific tenders in a specific respective region. It makes it easier for contractors to search for contracts, as they can use the NUTS classification to narrow down the area more easily.

- The attribute *hatArtDesÖffentlichenAuftraggeber* specifies a classification of the public client. The classification is defined as a character string and is therefore assigned to the attribute *hatArtDesÖffentlichenAuftraggebers* as a string data type.
- The attribute *hatHaupttätigkeiten* activities specifies the contracting authority's main activities, such as "Public safety and order", as a character string; we therefore assign this attribute the data type string.
- The attribute *überschreitetSchwellenwert* indicates whether the contract value reaches the threshold value for the application of European public procurement law. The Boolean data type is suitable for specifying whether the threshold value has been reached.
- The attribute *hatVergabeanwalt* indicates whether the contracting authority has appointed a procurement lawyer for the procurement procedure. The specification is made using the Boolean data type, as the only information it requires is whether or not legal advice from a public procurement lawyer exists for the award procedure.
- The attribute *istKomplexesVergabeverfahren* indicates whether the procurement procedure is classified as a complex procurement procedure. The specification is made using the Boolean data type, as the award procedure's complexity can be classified using a truth value.
- The attribute *istBundesausschreibung* specifies whether the invitation to tender for a service is relevant for the entire Federal Republic of Germany and is therefore independent of a federal state. The specification is made using the Boolean data type.
- The attribute *hatArtDesAuftraggebers* specifies the type of client, as in the attribute *hatArtDesÖffentlichenAuftraggebers*, and thus enables the identification of the requested service's specific users by naming the corresponding authority or organization with a security task. The string data type is used to store this information.
- The attribute *hatTCVPlan* specifies the planned contract value from the perspective of a potential bidding contractor. This value can be higher or lower than the client's estimated contract value (expressed by the attribute *hatGeschätztenWert*). We assign the attribute *hatTCVPlan* to the integer data type, as it is an integer that expresses the planned contract value.

- The attribute *betrifftBundesland* specifies which federal state is affected by this tender. The specification is made using a string, as the federal state's name is a character string. The assignment to a federal state can be automated using the attributes *hatNUTSCode* and *hatErfüllungsOrt* in such a way that a SWRL rule is created that automatically fills this attribute with the values from *hatNUTSCode* and *hatErfüllungsOrt*.

A complete description of the attributes of the safety-critical IT project ontology can be found in SETHUPATHY (2024), pp. 267–291.

3.2.3.7 Construction of cardinalities

We describe the construction of the cardinalities of non-taxonomic relations and attributes below, using a tabular representation structured as follows:

Domain	Name	Range	Cardinality
Vergabeverfahren	hatLose	Boolean	only

Table 41: Table structure for the explanation of cardinalities

Table 41 shows an example of a cardinality that specifies that an individual of the class *Vergabeverfahren* has exactly one (“only”) attribute value of the attribute *hatLose*, which is assigned to the Boolean data type. A complete overview of all cardinalities of both the non-taxonomic relations and the attributes of the safety-critical IT project ontology can be found in SETHUPATHY (2024), pp. 681–697 and pp. 698–722.

The cardinalities are constructed in Protégé in the “Sub-Class Of” field in the class area by clicking on the “Add” field, as shown in Figure 51 below.

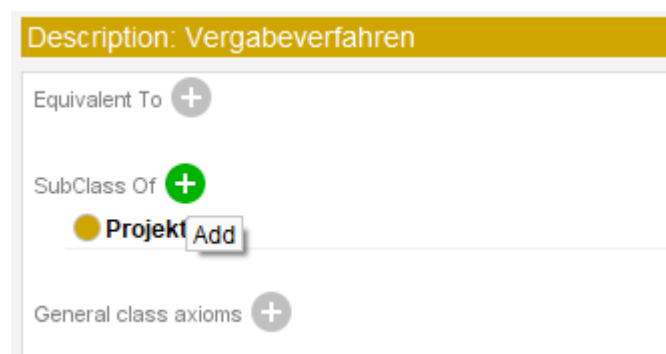


Figure 51: Calling up the function „SubClass Of“

The cardinality is entered in the Class Expression Editor by specifying the cardinality of a property (attribute or non-taxonomic relation). The previously mentioned example—that the class *Vergabeverfahren* for the attribute *hatLose* has exactly one attribute value with the Boolean data type—is expressed in the Class Expression Editor as follows:

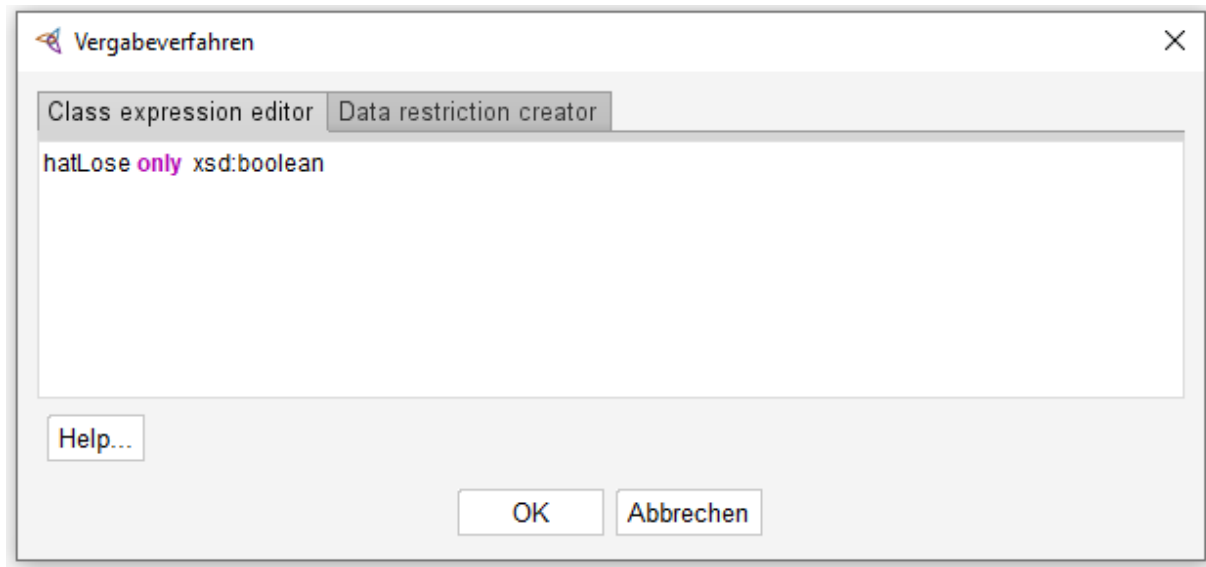


Figure 52: Specification of cardinality in the Class Expression Editor

Protégé offers various cardinalities, which we have illustrated using the example of the non-taxonomic relation *hatAusschlusskriterium*. The non-taxonomic relation *hatAusschlusskriterium* is assigned to the class *Vergabeverfahren* in the domain and to the class *Ausschlusskriterium* in the range.

Cardinality	Explanation
some	Each individual assigned to the class <i>Vergabeverfahren</i> has at least one individual of the class <i>Ausschlusskriterium</i> for the relation <i>hatAusschlusskriterium</i> .
only	Each individual assigned to the class <i>Vergabeverfahren</i> has exactly one individual of the class <i>Ausschlusskriterium</i> for the relation <i>hatAusschlusskriterium</i> .
min X	Each individual assigned to the class <i>Vergabeverfahren</i> has at least X individuals of the class <i>Ausschlusskriterium</i> for the relation <i>hatAusschlusskriterium</i> . The cardinality “min 1” is equivalent to the cardinality “some”.
exactly X	Each individual assigned to the class <i>Vergabeverfahren</i> has exactly X individuals of the class <i>Ausschlusskriterium</i> for the relation <i>hatAusschlusskriterium</i> . The cardinality “exactly 1” is equivalent to the cardinality “only”.

max X	Each individual assigned to the class <i>Vergabeverfahren</i> has a maximum of X individuals of the class <i>Ausschlusskriterium</i> for the relation <i>hatAusschlusskriterium</i> .
-------	---

Figure 53: Explanation of cardinalities

The cardinalities for the properties are justified on the basis of the class *Vergabeverfahren*. Figure 54 below shows the class *Vergabeverfahren* with its properties—attributes and non-taxonomic relations—as well as the assigned cardinalities.

Vergabeverfahren: Klasse
hatCPVCode: only String wurdeBereitsAufgehoben: only Boolean hatLinkZuBekanntmachung: min 1 String hatVorgeschaltetenTeilnahmeantrag: only Boolean hatErfüllungsOrt: min 1 String hatArtDesAuftrags: only String hatLose: only Boolean hatGeschätztenWert: min 1 Integer hatBezeichnungDesAuftrags: only String hatLaufzeitDerAngefragtenLeistung: min 1 Integer hatNUTSCode: some String hatArtDesÖffentlichenAuftraggeber: only String hatHaupttätigkeiten: only String überschreitetSchwellenwert: only Boolean hatVergabeanwalt: only Boolean istKomplexesVergabeverfahren: only Boolean istBundesausschreibung: only Boolean hatArtDesAuftraggebers: only String hatTCVPlan: only Integer betrifftBundesland: only Boolean hatVergabeverfahrenArt: only VergabeverfahrenArt hatAusschlusskriterium: min 1 Ausschlusskriterium hatZuschlagskriterium: min 1 Zuschlagskriterium hatEignungskriterium: min 1 Eignungskriterium hatVergabeverfahrenEntscheidung: min 2 VergabeverfahrenEntscheidung

Figure 54: Cardinalities of the attributes and
the non-taxonomic relations of the class *Vergabeverfahren*

The class *Vergabeverfahren* includes the following attributes: *hatCPVCode*, *wurdeBereitsAufgehoben*, *hatLinkZuBekanntmachung*, *hatVorgeschaltetenTeilnahmeantrag*, *hatErfüllungsOrt*, *hatArtDesAuftrags*, *hatLose*, *hatGeschätztenWert*, *hatBezeichnungDesAuftrags*, *hatLaufzeitDerAngefragtenLeistung*, *hatNUTSCode*, *hatArtDesÖffent*

lichenAuftraggebers, *hatHaupttätigkeiten*, *überschreitetSchwellenwert*, *hatVergabeanwalt*, *istKomplexesVergabeverfahren*, as well as *istBundesausschreibung*, *hatArtDesAuftragsgebers*, *hatTCVPlan*, *betrifftBundesland*.

The class *Vergabeverfahren* includes the following non-taxonomic relations: *hatVergabeverfahrenArt*, *hatAusschlusskriterium*, *hatZuschlagskriterium*, *hatEignungskriterium*, and *hatVergabeverfahrenEntscheidung*.

In the following, we first discuss the cardinalities of the attributes, then the cardinalities of the non-taxonomic relations of the class *Vergabeverfahren*.

- The attribute *hatCPVCode* is specified with the cardinality “only String”, as it is a unique identification number that is assigned only once and uniquely for an award procedure.
- The *hatNUTSCode* attribute is given the cardinality “some String”, as several geographical areas can be specified for each tender using the NUTS code. According to the rules for European calls for tender, the specification of a CPV code and a NUTS code is mandatory. To ensure that this requirement is met, a SWRL rule will be constructed later on. This rule specifies that the specification of a NUTS code, expressed by the attribute *hatNUTSCode*, is required if a CPV code, expressed by the attribute *hatCPVCode*, has been specified.
- The attribute *wurdeBereitsAufgehoben* is given the cardinality “only Boolean” to indicate that there can only be one Boolean value. This decision is justified by the fact that the indication of whether the award procedure has already been canceled in the past and re-tendered is sufficiently represented by a single Boolean value.
- The attribute *hatLinkZuBekanntmachung* is specified with the cardinality “min 1 String”. This is because there must be at least one link to the contract award notice. However, it is possible that the notice has been published on several procurement websites, setting the attribute’s cardinality to “min 1 String”.
- The attribute *hatVorgeschalteteTeilnahmeantrag* is specified with the cardinality “only Boolean”. An award procedure can only have exactly one preceding request to participate; it will not include multiple requests to participate. Therefore, we have selected the cardinality “only Boolean” for this attribute.
- The attribute *hatErfüllungsOrt* is specified with the cardinality “min 1 String”, as there must be at least one place of performance for an award procedure, but the service can also be provided at several locations. This can be the case, for example, if an award procedure is tendered in different lots.

- The attribute *hatArtDesAuftrag* is specified with the cardinality “only String”, as there can only be one designation for the type of order.
- The *hatLose* attribute is specified with the cardinality “only Boolean”, as a single specification of a Boolean value is sufficient to express whether the award procedure has been divided into several lots.
- The attribute *hatGeschätztenWert* is specified with the cardinality “min 1 Integer”, because the awarding authority must give at least one estimated value per award procedure in order to be able to decide whether a lower or higher threshold must be tendered. However, further estimated values may also be available if the award procedure has several lots.
- The attribute *hatArtDesÖffentlichenAuftraggeber* is specified with the cardinality “only String”, as exactly one authority inviting tenders for the service is named.
- The attribute *hatHaupttätigkeit* is constructed with the cardinality “only string”, since the announcements mention only exactly one main activity.
- The attribute *ueberschreitetSchwellenwert* is constructed with the cardinality “only Boolean”. The information as to whether a specific award procedure exceeds the threshold must only be entered once.
- The attribute *hatVergabeanwalt* is constructed with the cardinality “only Boolean”. It expresses whether an awarding attorney has been added for the award procedure. In this case, a Boolean value is sufficient to provide this linguistic means of expression for safety-critical IT projects.
- The attribute *istKomplexesVergabeverfahren* is constructed with the cardinality “only Boolean”. It is a one-off indication of whether the award procedure is classified as a complex award procedure and is required for subsequent automated classification using SWRL rules.
- The attribute *istBundesausschreibung* is constructed with the cardinality “only Boolean”, as a single specification is sufficient to indicate whether the award procedure is related to the federal level or to a specific federal state. It is used later to derive this attribute from the NUTS codes (attribute: *hatNUTSCode*) using SWRL rules.
- The attribute *betrifftBundesland* is constructed with the cardinality “only String”, as exactly one federal state is defined as being affected by the tendered service for an award procedure. In the case of a federal tender, the value “Germany” is

assigned to show that all federal states are affected. This assignment is also automated later on by a SWRL rule.

- The non-taxonomic relation *hatAusschlusskriterium* is constructed with the cardinality “min 1 *Ausschlusskriterium*”.
- The non-taxonomic relation *hatZuschlagskriterium* is constructed with the cardinality “min 1 *Zuschlagskriterium*”. For example, price can be considered an award criterion in an award procedure if it is selected as the exclusive award criterion. At least one criterion must be present in order to be able to make an award.
- The non-taxonomic relation *hatEignungskriterien* is constructed with the cardinality „min 1 *Eignungskriterium*”.
- The non-taxonomic relation *hatVergabeverfahrenArt* is constructed with the cardinality “only *VergabeverfahrenArt*”. This construction is justified by the fact that exactly one award type is intended for an award procedure.
- The non-taxonomic relation *hatVergabeverfahrenEntscheidung* is constructed with the cardinality “min 2 *VergabeverfahrenEntscheidung*”. This construction allows at least two decisions to be made in the case of a direct award, namely which type of award procedure is to be used and who will be awarded the contract.

3.2.3.8 Construction of Semantic Web Rules

We describe the construction of Semantic Web Rules below, presenting them in a table structured as follows:

Rule	Natural language translation

Table 42: Table structure for the explanation of semantic web rules

In the “Rule” column, a Semantic Web Rule (SWR) is displayed in the Protégé language. In the “Natural language translation” column, the rule content is explained in natural language.

The Semantic Web Rule Language (SWRL) is the language in which the following rules are defined. It is a combination of OWL DL and Unary / Binary Datalog RuleML. A rule consists of an antecedent condition (antecedent) and a consequent: “antecedent $\rightarrow$ consequent”.

Two expressions (atoms) that follow each other in a SWRL rule are separated by the character “^”. The antecedent and consequent parts of a rule can consist of 0 to n expressions, whereby each expression can be represented in the following form: $C(x)$, $P(x, y)$, $contains(x, y)$, $greaterthan(x, y)$. These forms are to be understood as follows: C is a class, P is a property, x and y are variables or can represent individuals, $contains(x, y)$ and $greaterthan(x, y)$ are auxiliary functions that are predefined in SWRL.

A rule is triggered exactly when every expression in the antecedent condition is fulfilled. If this happens, the consequence follows. We explain such a rule below using an example consisting of the following expressions:

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

Expressions that begin with a “?”—such as “?its”—refer to the individuals of the classes. In this example, the variable “?its” refers to all individuals of the class *SicherheitskritischesITSystem*.

The rule is constructed as follows:

Antecedent condition	Consequence
SicherheitskritischesITSystem(?its) ^ hatSchutzbedarfskategorie (?its, ?s) ^ swrlb:contains(?s, „sehr hoch“) ^ MitProjektbezugMitarbeiter(?p)	hatErweiterteSicherheitsueberpruefung(?p, true)

Table 43: Presentation of an exemplary SWRL rule

This rule expresses by way of example that the project employees must have an extended security check if a security-critical IT system has a very high protection requirement category: If a safety-critical IT system has the protection requirement category “sehr hoch” and individuals exist for the class *MitProjektbezugMitarbeiter*, then the property *hatErweiterteSicherheitsueberpruefung* of the class *MitProjektbezugMitarbeiter* is set to “true”.

The use of Semantic Web Rules in Protégé requires the use of the Pellet Reasoner. If a different reasoner is used, an error message is displayed. The use of the Pellet Reasoner for Semantic Web Rules has been suggested, for example, by DEBELLIS (2021), p. 15, SYCHEV/ANIKIN/DENISOV (2021), p. 472, and BATSAKIS/TACHMAZIDIS/ANTONIOU (2017), p. 25. Apache Jena as a reasoner for SWRL can be seen as more advantageous in terms of runtime and the use of different programming languages; cf. SYCHEV/ANIKIN/DENISOV (2021), p. 480.

Semantic Web Rules Language is constructed using a plug-in in Protégé. The construction aid “SWRL-Tab” is called up via Window>Tabs>SWRLTab. The following window is opened in Figure 55 to create a Semantic Web Rule.

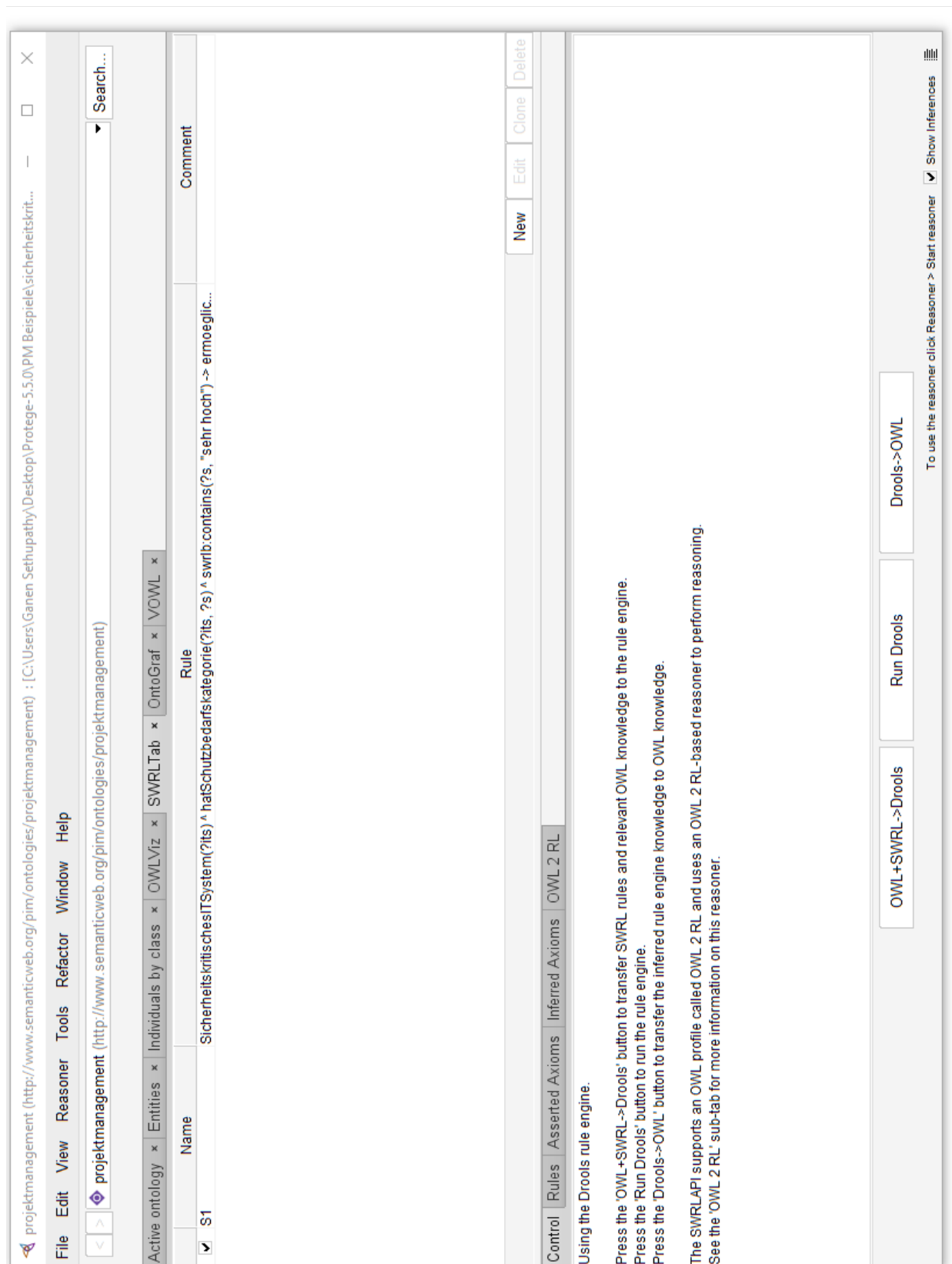


Figure 55: SWRL tab in Protégé

The construction of a Semantic Web Rule requires taking into account different expressions for the antecedent condition and for the consequence, as shown in the following rule as an example.

Rule	Natural language translation
<code>SicherheitskritischesITSystem(?its) ^ hatSchutzbedarfskategorie(?its, ?s) ^ swrlb:contains(?s, „sehr hoch“) → ermoglichtNearshore(?its, false) ^ ermoglichtOffshore(?its, false)</code>	If a safety-critical IT system is categorized as “very high”, then neither nearshore nor offshore is possible.

Table 44: SWRL rule with its natural language translation

First, the class *SicherheitskritischesITSystem* is declared with the variable “?its”, so that all individuals of the class *SicherheitskritischesITProjekt* are to be taken into account in the expression “*SicherheitskritischesITProjekt*(?its)”. This expression is bound to the expression “*hatSicherheitsstufe*(?its, ?s)” using “^”. The expression “*hatSicherheitsstufe*(?its, ?s)” refers to the attribute *hasSafetyLevel* of the *SicherheitskritischesITProjekt*. The variable “?s” stands for the attribute *hatSicherheitsstufe* and is checked for the value “very high” using the predefined auxiliary function *swlb:contains*. If the antecedent condition of the rule is fulfilled, the attributes *ermoglichtNearshore* and *ermöglicht Offshore* are set to “false” as a consequence of the rule.

The Semantic Web Rule described above represents the rule that no development work may be performed outside of Germany for a security-critical IT system if its protection requirement category has been classified as “very high”.

The previously explained construction of a Semantic Web Rule takes place in the SWRL tab of Protégé, as shown in Figure 56 below.

Figure 56: Construction of a Semantic Web Rule in Protégé

If the design is incorrect, the “Ok” button remains grayed out so that confirmation is not possible. Furthermore, an error message is displayed in the “Status” field so that an error analysis can take place. If the design is error-free, “Ok” is displayed in the “Status” field, as shown in Figure 56.

Below, we explain the construction of the Semantic Web Rules for the class *Vergabeverfahren*. The rule shown in Table 45 states that the threshold is reached if the award’s estimated value is greater than or equal to 250,000 € (“euros”). This modeling makes it possible to define a rule based on the estimated value as to whether the award is in the below-threshold or above-threshold range. When constructing this rule, the auxiliary function *swrlb:greaterThanOrEqual* is used, which performs the case check for the integer value 250,000.

Rule	Natural language translation
$\text{Vergabeverfahren}(\text{?v}) \wedge$ $\text{hatGeschaetztenWert}(\text{?v}, \text{?np}) \wedge$ $\text{swrlb: greaterThanOrEqual}(\text{?np}, 250000)$ $\rightarrow \text{ueberschreitetSchwellenwert}(\text{?v}, \text{true})$	If the estimated value of a procurement award procedure reaches 250,000 €, the threshold value is reached.

Table 45: Rule for checking the threshold value

Table 46 below shows the rule that uses the NUTS code to determine the federal state affected by the tender. This rule’s construction enables the automated assignment of the federal state and the indication that the tender is not a federal tender. This design decision is justified by the fact that the uniqueness of the NUTS code enables an automated

assignment of the federal state. This also applies to several NUTS codes, if multiple federal states are involved.

Rule	Natural language translation
$\text{Vergabeverfahren}(?v) \wedge$ $\text{hatNutsCode}(?v, ?nc) \wedge$ $\text{swrlb:contains}(?, \text{DE1})$ $\rightarrow \text{betrifftBundesland}(?v, \text{„Baden-Württemberg“})$ $\wedge \text{istBundesausschreibung}(?v, \text{„false“})$	If an award procedure contains the NUTS code “DE1” as a substring, then the attribute <i>betrifftBundesland</i> is set to the value “Baden-Württemberg” and the attribute <i>istBundesausschreibung</i> is set to “false.”

Table 46: Rule for automated derivation of the federal state

The following Table 47 illustrates the rule according to which the name of the required service regulation for the safety-critical IT project can be automatically derived from the attribute *hatArtDesAuftraggebers* of an award procedure. This applies, for example, if a substring such as “fire department” is present as a value in the attribute *hatArtDesAuftraggebers*.

Rule	Natural language translation
$\text{Vergabeverfahren}(?v) \wedge$ $\text{SicherheitskritischesITProjekt}(?its) \wedge$ $\text{erfordertVergabeverfahren}(?its, ?v) \wedge$ $\text{hatArtDesAuftraggebers}(?v, ?ha) \wedge$ $\text{swrlb:contains}(?ha, \text{„Feuerwehr“}) \wedge$ $\text{erfordertDienstvorschrift}(?its, ?a)$ $\rightarrow \text{hatBezeichnung}(?a,$ $\quad \text{„Feuerwehrdienstvorschrift“})$	If an award procedure involves a safety-critical IT project in which the fire department is specified as the type of client (<i>hatArtDesAuf Auftraggebers</i>) and a specific service regulation (<i>erfordertDienstvorschrift</i>) is required, this service regulation must bear the designation “Feuerwehrdienstvorschrift” (<i>hatBezeichnung</i>).
$\text{Vergabeverfahren}(?v) \wedge$ $\text{SicherheitskritischesITProjekt}(?its) \wedge$ $\text{erfordertVergabeverfahren}(?its, ?v) \wedge$ $\text{hatArtDesAuftraggebers}(?v, ?ha) \wedge$ $\text{swrlb:contains}(?ha, \text{„Polizei“}) \wedge$ $\text{erfordertDienstvorschrift}(?its, ?a)$ $\rightarrow \text{hatBezeichnung}(?a,$ $\quad \text{„Polizeidienstvorschrift“})$	If a procurement procedure concerns a security-critical IT project in which the police is specified as the type of contracting authority (<i>hatArtDesAuftraggebers</i>) and a specific service regulation (<i>erfordertDienstvorschrift</i>) is required, this service regulation must bear the designation “Police Service Regulation” (<i>hatBezeichnung</i>).

Table 47: Rules for the automated derivation of a service regulation

Table 48 below shows a rule that uses the NUTS code to classify the relevance of the award procedure as Germany-wide. As the NUTS code for Germany is always constructed with “DE” and two further digits for the federal states, it is necessary at this point to identify whether only the term “DE” is mentioned, with no further combination of digits. This is checked using the auxiliary function “swrlb:endsWith(?nc, „DE“) $\wedge$ swrlb:startsWith(?nc, „DE“). The auxiliary function “swrlb:startsWith(?nc, „DE“)” checks whether the string begins with “DE”, and the auxiliary function “swrlb:endsWith(?nc, „DE“)” checks whether the string ends with “DE”. Both auxiliary functions are connected with a logical “and”. If this is the case, the attribute *betrifftBundesland* is set to the value “Deutschland” and the attribute *istBundesausschreibung* is set to “true”, to make it clear that the award procedure is a federal tender.

Rule	Natural language translation
Vergabeverfahren(?v) $\wedge$ hatNutsCode(?v, ?nc) $\wedge$ swrlb:contains(?nc, „DE“) $\wedge$ swrlb:endsWith(?nc, „DE“) $\wedge$ swrlb:startsWith(?nc, „DE“) $\rightarrow$ betrifftBundesland(?v, „Deutschland“) $\wedge$ istBundesausschreibung(?v, „true“)	If an award procedure contains the NUTS code “DE” as a substring and only the term “DE”, then the property “betrifftBundesland” is set to “Deutschland” and the property “istBundesausschreibung” is set to “true”.

Table 48: Rule for the automated derivation of a federal tender

Table 49 below contains a rule that is based on the properties (attributes and non-taxonomic relations) of an allocation procedure, namely:

- ueberschreitetSchwellenwert
- hatExterneVergabeanwalt
- wurdeBereitsAufgehoben
- hatGeschaetztenWert
- hatVergabekostenPlan

These determine whether the procurement procedure is a complex procurement procedure.

The selection of characteristics for the interpretation of whether a procurement procedure is complex or not proves to be debatable. In this article, we have selected the aforementioned award procedure characteristics for the following reasons:

- If the threshold value is exceeded, comprehensive principles of public procurement law must be taken into account.
- If the awarding authority decides to involve an external procurement lawyer for the award procedure, it can be assumed that more complex issues of public procurement law need to be clarified.
- If an award procedure has already been canceled in the past for various reasons, this could indicate its complexity (for example, cancellation due to a complaint from an unsuccessful bidder).
- If the award amount exceeds 500,000 €, it exceeds the average award value of a supply or service contract (as of 2019) and is twice as high as the threshold value. (We do not include construction contracts in this analysis, as they are irrelevant for security-critical IT projects.)
- With planned award costs (costs incurred by the contracting authority as a result of the award procedure) of more than 50,000 €, it can be assumed that several consultations with the bidders are necessary in order to successfully conclude the award procedure. The award costs must take into account not just the external costs (e.g., for procurement law support), but also the internal ones (e.g., for coordination with the bidders).

The aim of the following rule is to use the aforementioned characteristics to classify whether the award procedure is complex.

rule	natural language translation
Vergabeverfahren(?v) ∧ ueberschreitetSchwellenwert(?v, „true“) ∧ hatExterneVergabeanwalt(?v, „true“) ∧ wurdeBereitsAufgehoben(?v, „true“) ∧ hatGeschaetztenWert(?v,?gw) ∧ swrlb:greaterThan(?gw, „500000“) ∧ hatVergabekostenPlan(?v,?vvp) ∧ swrlb:greaterThan(?vvp, „50000“) → istKomplexesVergabeverfahren (?v, „true“)	If an award procedure exceeds the threshold value, has an external award lawyer, has already been canceled in the past, has an estimated award amount of over 500,000 €, and its award costs are planned to be over 50,000 €, then it is to be classified as complex.

Table 49: Rule for automated determination of a complex tender

The following rule in Table 50 states that an award procedure has at least one award criterion if it includes at least one exclusion criterion. In other words, a procurement procedure must meet certain requirements in order to be considered.

Rule	Natural language translation
$\text{Vergabeverfahren}(?v) \wedge$ $\text{hatAusschlusskriterium}(?v, ?a)$ $\rightarrow \text{hatZuschlagskriterium}(?v, ?b)$	If an award procedure includes at least one exclusion criterion, then it must also have at least one award criterion.

Table 50: Rule for the automated determination of exclusion and award criteria

3.2.3.9 Construction of global individuals

We discuss the construction of global individuals below. Strictly speaking, it can be argued that individuals are not components of an ontology. Nevertheless, we address and regard global individuals as such because they are used in the safety-critical IT project ontology to express generally valid statements. Global individuals should not be changed in concrete ontology-supported CBR systems, nor should they be supplemented by attributes or relations. We here follow this view in order to construct global individuals and introduce them as ontology components.

Global individuals are characterized as follows in comparison to local individuals:

- Global individuals are individuals that are defined in the safety-critical IT project ontology and do not relate to a specific project, but apply across all projects.
- Global individuals can be defined as fixed categories (e.g., low, medium, high, very high).
- Global individuals provide linguistic means of expression for temporal constants that can be used for several projects simultaneously, such as specific locations and colors.
- Constants that are used repeatedly can be specified using global individuals and thus automatically recorded in an ontology-supported CBR system.

Global individuals are specified in the following table:

Individual name:	
Class:	
Property	Value

Table 51: Table structure for the specification of global individuals

For each global individual, the tabular display includes the individual name, the class to which the individual belongs, the individual-describing properties (attributes and non-taxonomic relations), and the individual's associated values.

Only global individuals are created in Protégé. Protégé offers various options for creating (global) individuals. The most common methods are in the areas:

- “Individuals by class”
- “Entities” under the tab “Classes”
- “Entities” under the tab “Individuals”

The following is an example of the latter method in the “Entities” area under the “Individuals” tab. Figure 57 below shows the “Individuals” tab in the “Entities” area.

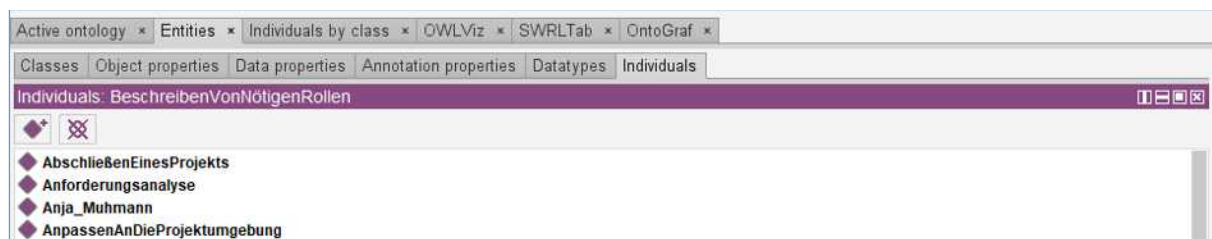


Figure 57: Overview of the available individuals

A new individual is created by selecting “Add Individual”.

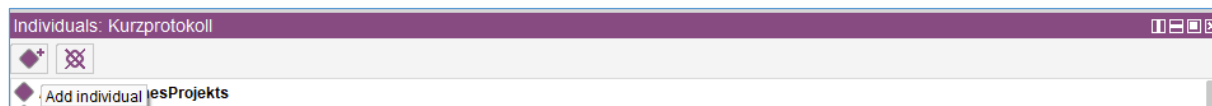


Figure 58: Adding a new individual

The individual's name is then specified, as shown in Figure 59 below.

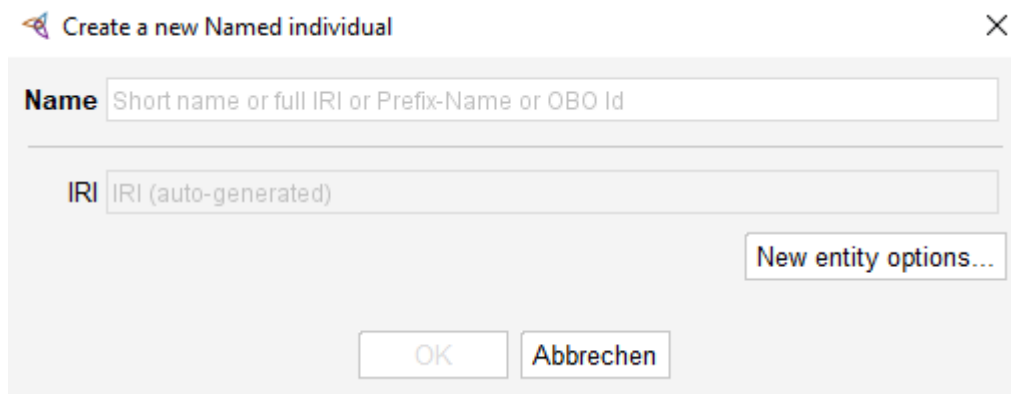


Figure 59: Name of the individual

As soon as the individual's name—here *RisikoIdentifizieren*—has been created, it can be found in the list of all individuals and be assigned to a class. See Figure 60 below.

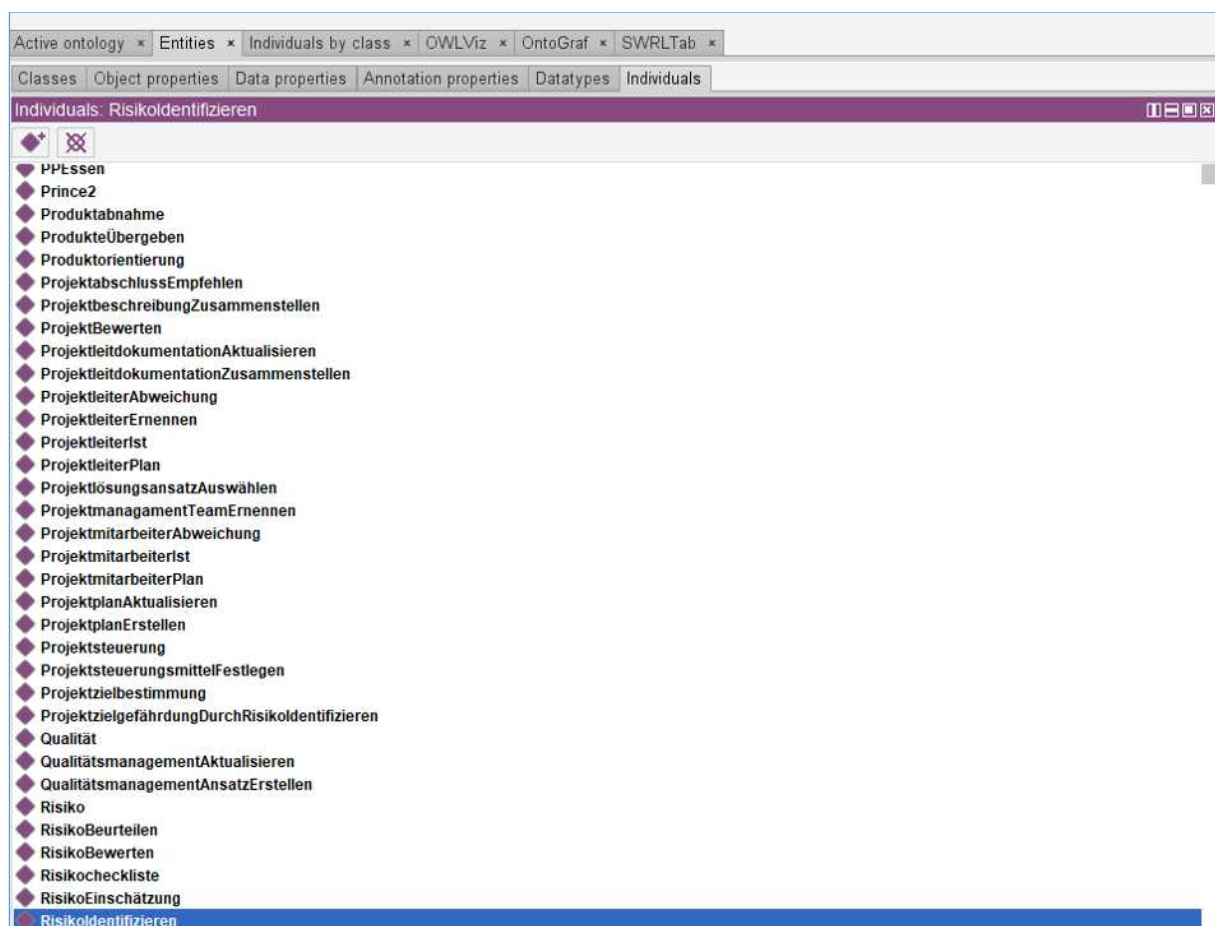


Figure 60: Display of the newly created individual *RisikoIdentifizieren* in the overview

To assign the individual *RisikoIdentifizieren* to a class, open the “Types” area of the Class Expression Editor and select the relevant class for the individual.

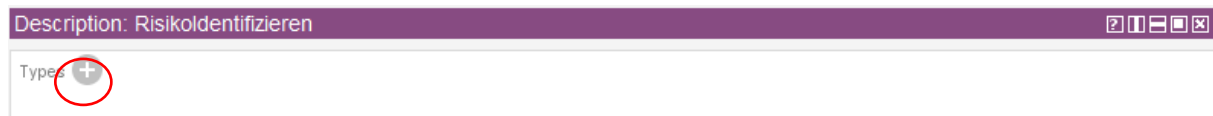


Figure 61: Assignment of a class for the individual *RisikoIdentifizieren* in the Class Expression Editor

In the example considered here, the class *RisikomanagementProzess* is selected for the individual *RisikoIdentifizieren*. This means that the individual *RisikoIdentifizieren* is an individual of the class *RisikomanagementProzess*, as shown in Figure 62 below.

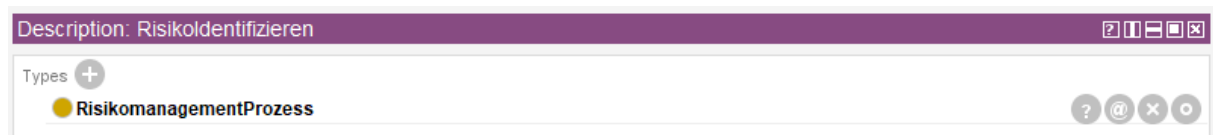


Figure 62: Assignment of the class *RisikomanagementProzess* to the individual *RisikoIdentifizieren*

After the class *RisikomanagementProzess* has been assigned, its properties must be assigned to the individual *RisikoIdentifizieren*. The class *RisikomanagementProzess* has a non-taxonomic relation *beinhaltetMaßnahme* with the cardinality “*min 1 Maßnahme*”, so that at least one measure must be assigned. The measures *KontextIdentifizieren* and *GefährdungIdentifizieren* are assigned to the individual *RisikoIdentifizieren*, as shown in Figure 63 below

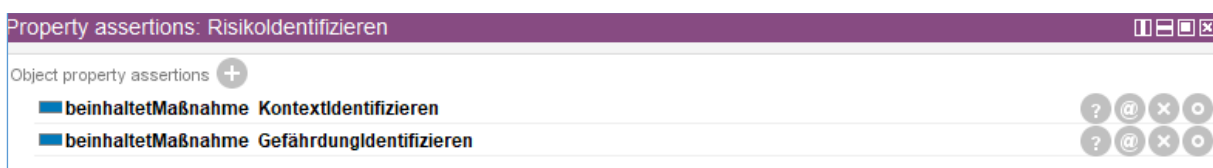


Figure 63: Assigned properties for the individual *RisikoIdentifizieren*

The measures *KontextIdentifizieren* and *GefährdungIdentifizieren* are also individuals, belonging to the class *Maßnahme*.

Figure 64 below summarizes the construction of the individual *RisikoIdentifizieren*. The dashed line illustrates the non-taxonomic relationship at the individual level, which is already represented by the solid line at the class level.

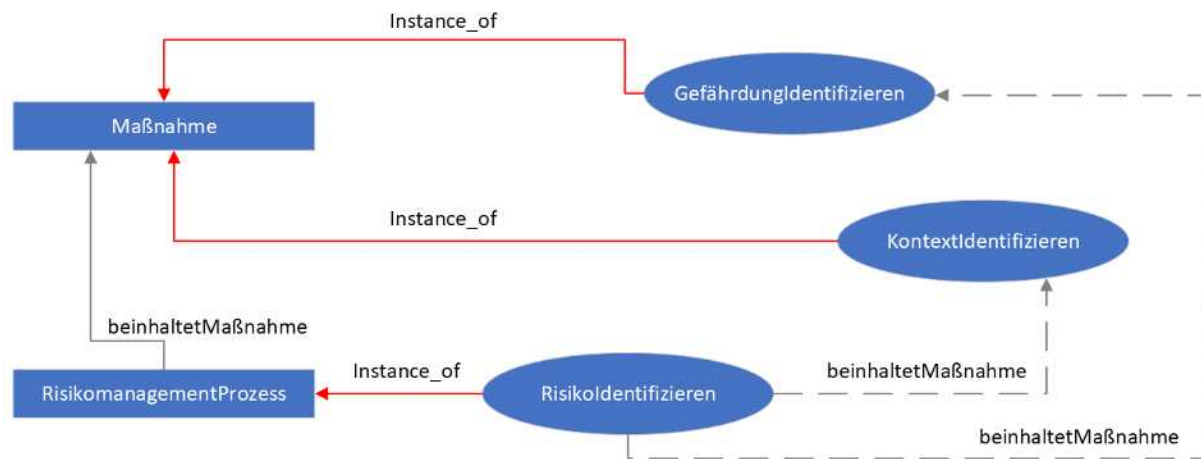


Figure 64: Individuals *RisikoIdentifizieren*, *GefährdungIdentifizieren*, und *KontextIdentifizieren* in the context of their classes

In the following, the individuals *PRINCE2* and *RisikomanagementPRINCE2* are explained by way of example with regard to their construction.

Table 52 below contains the individual *PRINCE2*, which belongs to the class *Projektmanagementmethode*.

Individual: PRINCE2	
Class: Projektmanagementmethode	
Property	Value
bestehtAusThema	Änderung
bestehtAusThema	Risiko
bestehtAusThema	Qualität
bestehtAusThema	BusinessCase
bestehtAusThema	Fortschritt
bestehtAusThema	Organisation
bestehtAusThema	Plan
bestehtAusGrundprinzip	LernenAusErfahrung
bestehtAusGrundprinzip	SteuernNachDemAusnahmeprinzip

bestehtAusGrundprinzip	FortlaufendeGeschäftlicheRechtfertigung
bestehtAusGrundprinzip	ProduktOrientierung
bestehtAusGrundprinzip	AnpassendieProjektumgebung
bestehtAusGrundprinzip	SteuerungÜberManagementphase
bestehtAusGrundprinzip	DefinierteRolleUndVerantwortlichkeit
bestehtAusProzess	SteuernEinerPhase
bestehtAusProzess	VorbereitenEinesProjektes
bestehtAusProzess	InitierenEinesProjektes
bestehtAusProzess	AbschließenEinesProjektes
bestehtAusProzess	LenkenEinesProjektes
bestehtAusProzess	ManagenEinesPhasenübergangs
bestehtAusProzess	AbschließenEinesProjektes

Table 52: Representation of the individual *PRINCE2*

The individual *PRINCE2* has the properties *consistsOfTopic*, *consistsOfPrinciple*, and *consistsOfProcess*, each with the cardinalities “exactly 7”. The individual *PRINCE2* can therefore be associated with the seven basic principles, the seven topics, and the seven processes from the PRINCE2 project management method (the “framework”); cf. AX-ELOS (2015), pp. 34–36, 47–59, and 135–140.

Figure 65 illustrates the automated construction of the *PRINCE2* global individual in an ontology-supported CBR system constructed using the CBR tool jCORA. This individual is defined via the non-taxonomic relation *bestehtAusProzess* with the individual *AbschließenEinesProjektes* connected. The latter is in turn linked via the non-taxonomic relation *beinhaltetMaßnahme* to other individuals, for example with the individual *Produktabnahme*. In this way, the measures defined in PRINCE2 for the process “Closing a Project” can be expressed.



Figure 65: The exemplary representation of global individuals using the example of the individual *PRINCE2* in jCORA

Table 53 below shows the individual *RisikomanagementPRINCE2* with the associated properties.

Individual: RisikomanagementPRINCE2	
Class: Risikomanagement	
Property	Value
bestehtAusRisikomanagementProzess	RisikoBewerten
bestehtAusRisikomanagementProzess	RisikoMaßnahmenImplementieren
bestehtAusRisikomanagementProzess	RisikoIdentifizieren
bestehtAusRisikomanagementProzess	RisikoPlanen

Table 53: Representation of the individual *RisikomanagementPRINCE2*

The individual *RisikomanagementPRINCE2* has the non-taxonomic relation *bestehtAusRisikomanagementProzess* with the cardinality “exactly 4” for the class *RisikomanagementProzess* in the relation’s range. The individual *RisikomanagementPRINCE2* is constructed according to PRINCE2’s risk management, which has the following individuals of the class *RisikomanagementProzess* as property values:

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

The risk management processes in PRINCE2 are based on the DEMING circle. The DEMING circle is also known as the PDCA cycle, DEMING cycle, or SHEWHART cycle (or “wheel” or “circle”); cf., e.g., KALS (2021), p. 287; ANGERMEIER (2016). The DEMING cycle generally describes an iterative four-stage process as a cycle for the continuous improvement of work processes and was introduced by the US statistician DEMING; cf. SYSKA (2006), p. 100. It can be applied in any domain, whereby the acronym PDCA stands for Plan, Do, Check, and Act and describes the cycle’s four processes. The central point of the DEMING cycle is that—regardless of how the individual process steps are labeled—the effects of a process step are reviewed and these findings in turn have a retroactive influence on the entire cycle’s management. Strictly speaking, the PRINCE2 risk management process has a further process step that is not included in the DEMING

cycle. This process step, referred to as “communicating risk”, takes place throughout the cycle; cf. AXELOS (2018), p. 124. As it does so in parallel with all risk management processes, we here define this process as a measure and not as a separate process step. This design decision does not contradict the PRINCE2 risk management process steps, as the list of process steps in AXELOS (2018), p. 124, also refers to four consecutive process steps.

We created the individual *RisikomanagementPRINCE2* using the underlying PRINCE2 and risk management ontology, then adapting the result for the security-critical IT project ontology. We essentially made these adjustments because the underlying PRINCE2 and risk management ontology provides properties for the individual *RisikomanagementPRINCE2* that are incorrectly classified as risk management processes. Take, for example, the individual *RisikoregisterDokumentieren*, which is defined as a risk management process in the original PRINCE2 and risk management ontology, but is merely a measure and should be provided as a property for all four risk management processes.

Figure 66 below shows the adapted individual *RisikomanagementPRINCE2* from the safety-critical IT project ontology, which is based on the DEMING cycle defined by PRINCE2. Each risk management process has additional measures as characteristics. Figure 66 shows an example of this for the risk management process *RisikoIdentifizieren*, which includes the measures *KontextIdentifizieren* and *GefährdungIdentifizieren*.

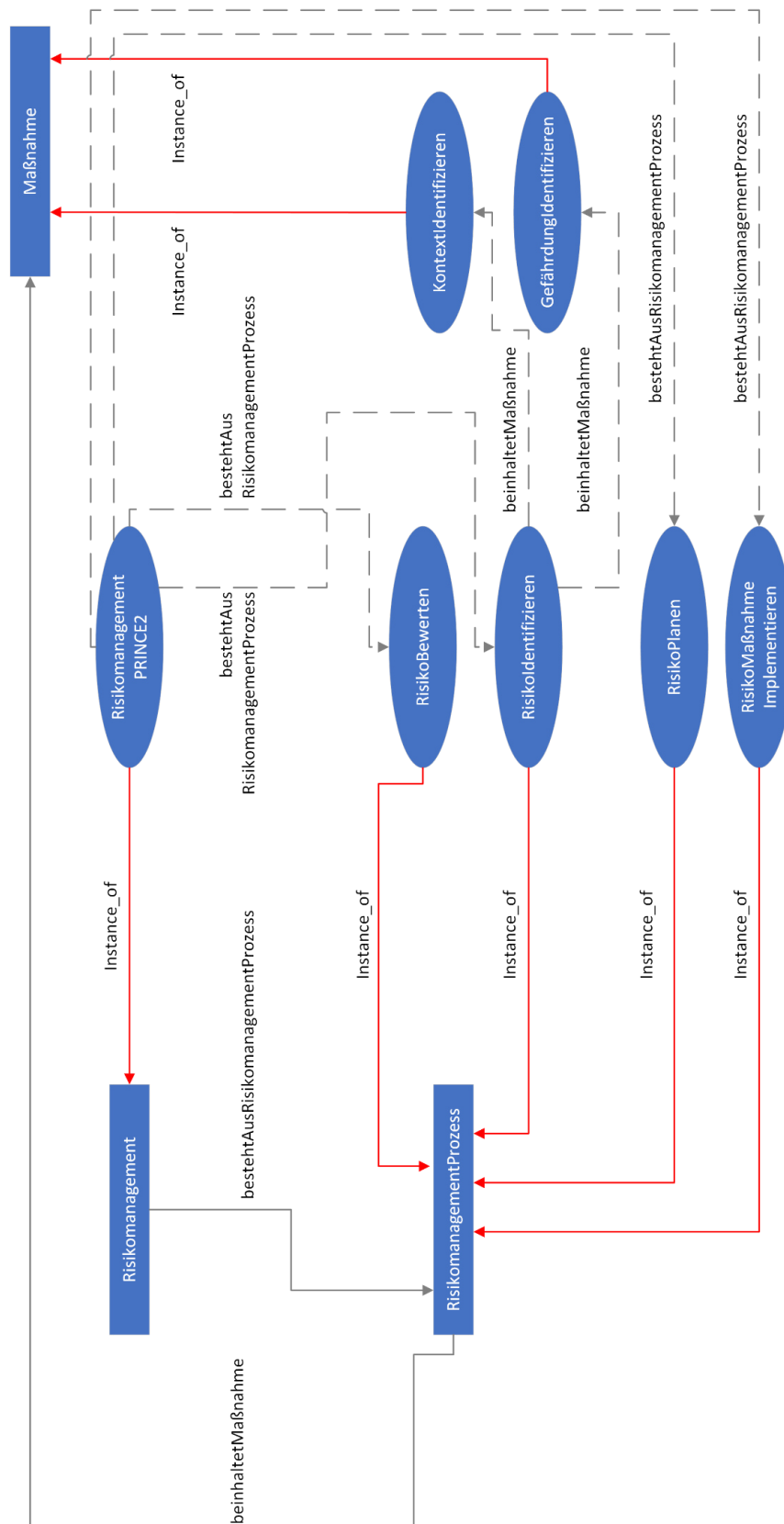


Figure 66: Risk management from PRINCE2 with the associated risk management processes

By using the individual *RisikomanagementPRINCE2* from the safety-critical IT project ontology, the PRINCE2 risk management processes can be constructed automatically in the CBR tool jCORA, as shown in Figure 67 below.

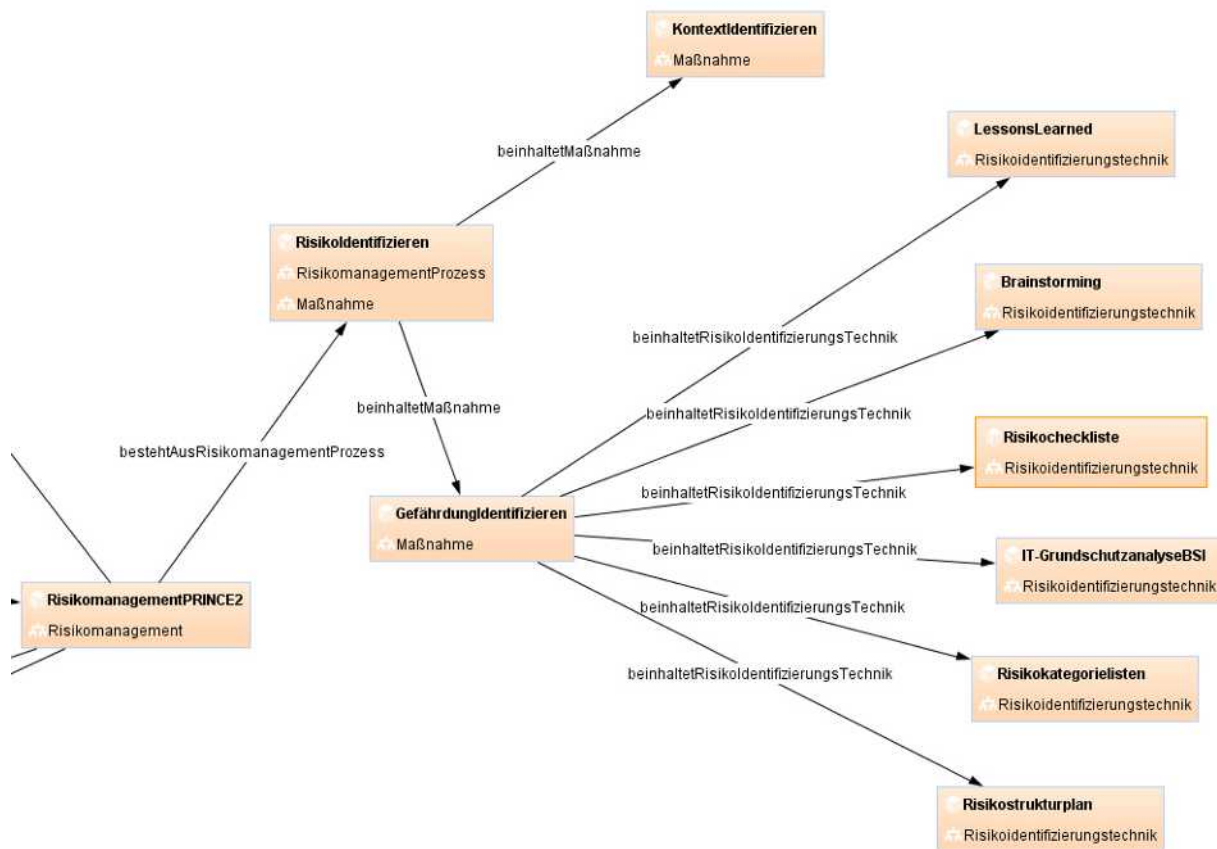


Figure 67: Representation of global individuals using the example of the individual *RisikomanagementPRINCE2* in jCORA

Figure 67 shows that the individual *RisikoIdentifizieren* (first line in its ontology-related node in bold) belongs to two classes (polymorphism), namely to both the class *Risiko managementprozess* and the class *Maßnahme* (second and third lines in its ontology-related node in normal print).

3.3 Case-based reasoning on the basis of a safety-critical IT project ontology

3.3.1 Ontology-supported case-based reasoning system using jCORA

3.3.1.1 Description of the prototype CBR tool jCORA

The CBR tool jCORA (“java-based Case- and Ontology-based Reasoning Application”) is a prototype software for the construction of ontology-based CBR systems. It was developed for the “intelligent” reuse of experience-based knowledge in operational project management. This CBR tool was designed and implemented at the Institute for Production and Industrial Production Management (PIM) at the University of Duisburg-Essen as part of the BMBF joint project OrGoLo and further developed as part of the BMBF joint project KI-LiveS. Cf. ZELEWSKI/HEEB/SCHAGEN (2022), pp. 225–251; BERGEN-RODT/KOWALSKI/ZELEWSKI (2015), pp. 475–541.

An essential prerequisite for the use of the CBR tool jCORA is the existence of an ontology, in this case the safety-critical IT project ontology. This need for an ontology means that jCORA is referred to as an ontology-supported CBR tool. In jCORA, ontologies are specified in the settings as a local path to an OWL file. Ontologies in OWL format are required as an exchange format for ontologies between an ontology editor and an ontology-supported CBR system. This is illustrated in Figure 68 below.

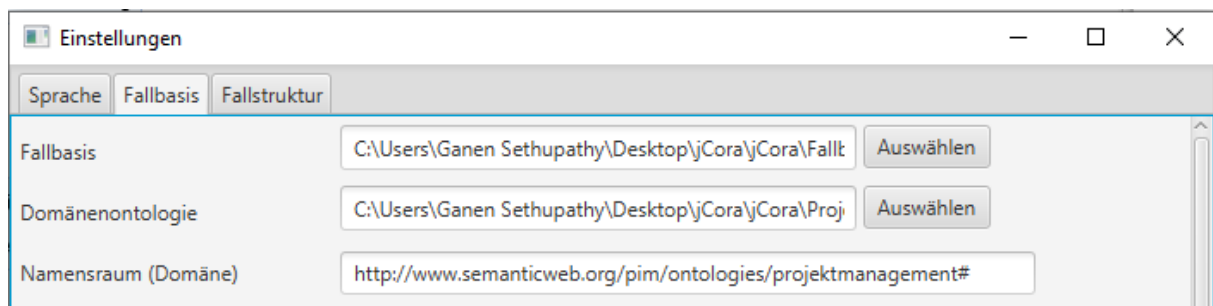


Figure 68: Settings in the CBR tool jCORA

The ontology specifies those linguistic means of expression that are considered necessary or at least helpful in order to represent the knowledge gained from safety-critical IT projects in jCORA in the present case.

The working area of jCORA lies among local, project-specific individuals and their individual values. The ontology editor Protégé, on the other hand, is limited to the underlying ontology and operates mainly at the class level, including the taxonomic relation “is_a” and other non-taxonomic relations between individuals. In addition, there are

global individuals that apply to the entire ontology across projects, as well as rules (often referred to as “axioms”), such as the Semantic Web Rules already presented.

The boundary between the class level and the individual level is not always easy to draw. This is due to the fact that individuals can be constructed both in an ontology editor, such as Protégé, and in a CBR tool, such as jCORA.

As already explained, *global individuals* constructed with Protégé do not refer exclusively to a single, project-specific reality section, but provide linguistic means of expression across projects, with which individuals from any project-related reality section can be modeled—e.g., cross-project management methods such as PRINCE2. In some publications, the individuals specified in an ontology are referred to as a global knowledge component.

In contrast, *local individuals* as local knowledge components refer exclusively to a single project. This difference is also made clear by the different areas of application of the CBR tool jCORA and the ontology editor Protégé: While jCORA works at the level of local, project-specific individuals and their associated values, Protégé focuses on the underlying ontology and operates mainly at the class level, including global, cross-project individuals. The question of whether an individual is constructed globally or locally can only be answered depending on the context.

In addition, the distinction between global and local individuals can impact the algorithm for calculating similarity in the CBR tool jCORA; cf. SETHUPATHY (2024), p. 324; ZELEWSKI/SCHAGEN (2022), p. 32. There it is shown that global individuals can lead to incorrect similarity calculations in jCORA if these *global* individuals, despite their character as global knowledge components, are used when specifying projects (cases)³ which can be extended in jCORA by *local* relation or attribute values. This represents a significant functional limitation due to an uncertainty of correctness in the practical use of the CBR tool jCORA.

The *case structure* is of paramount importance in an ontology-based CBR system. According to ASSALI/LENNE/DEBRAY (2010), p. 105; WATSON (2003), pp. 27–28; WATSON (1998), pp. 176–177, a distinction must be made between a homogeneous and a heterogeneous case structure of CBR tools. If every case considered in a CBR tool has the same classes, relations, and attributes, then it is referred to as a homogeneous case

3) In this article, we use the terms “cases” and “projects” synonymously, because we use case-based reasoning exclusively for the management of the reuse of *project-related* experience-based knowledge, so that no distinction is made between cases of case-based reasoning in general and projects in the special sense of (computer-aided) knowledge management for projects. In this article, the terms “cases” versus “projects” are used as they appear plausible in the respective current argumentation context.

structure. Otherwise, a heterogeneous case structure exists; cf. BERGENRODT/KOWALSKI/ZELEWSKI (2015), p. 491.

For the application of case-based reasoning, various specialist literature assumes that the case structure is known *ex ante*, is the same for all cases, and does not change in the course of using a CBR tool. This case structure is considered homogeneous—it makes it easier to compare the properties of different cases, and their similarities can be calculated using simple heuristics; cf. ASSALI/LENNE/DEBRAY (2010), p. 98. In a complementary way, it is emphasized that CBR tools with a heterogeneous case structure are considered more difficult to implement than CBR tools with a homogeneous case structure; cf. AVESANI/SUSI (2010), p. 183, ABOU ASSALI et al. (2009), p. 564. The particular challenge of heterogeneous case structures is seen in the context of determining similarity; cf. ASSALI/LENNE/DEBRAY (2010), p. 98. This assessment led to the development of the CBR tool jCORA, which is intended to enable the most flexible handling possible of heterogeneous case structures; cf. BERGENRODT/KOWALSKI/ZELEWSKI (2015), p. 479. The real complexity of security-critical IT projects in operational practice also suggests that projects have a heterogeneous case structure (project structure).

Knowledge structuring in the CBR tool jCORA is divided into three main *project-related* components. A distinction is made between:

- Project description
- Project solution
- Project evaluation

The CBR tool jCORA's functionality is based on the CBR cycle described earlier. Its focus lies on said cycle's *retrieve phase*, during which a similarity algorithm is used to determine the *similarity* between the description of a new project and the descriptions of old, already completed projects stored in the project knowledge base. This algorithm's basic features are based on the similarity algorithm of BEIBEL (2011), pp. 159–215. However, BEIBEL's similarity algorithm has various shortcomings that limit its suitability for practical use; cf. BERGENRODT/KOWALSKI/ZELEWSKI (2015), pp. 479–480. These include the following limitations in particular:

- The similarity algorithm does not support a recursive flow structure.
- The similarity algorithm has limitations with regard to the permissible cardinalities of non-taxonomic relations and attributes.
- The similarity algorithm does not support restrictions, conjunctions, and disjunctions or non-taxonomic relations with multiple values in the descendant area.

As part of the similarity calculation of two projects, the CBR tool jCORA can store weights for project characteristics that users consider to be relevant to similarity (we will come back to this later).

The CBR tool jCORA supports the *retain phase* without restriction and automatically. However, jCORA suffers from considerable limitations with regard to the CBR cycle's *reuse* and *revise phases*— as do most other currently known CBR tools:

- During the reuse phase, jCORA does not support automated adaptation to the project solution of an old project that is as similar as possible and has already been implemented. It only provides a copy function as a “zero adaptation”, which does not enable effective adaptation of project solutions from old to new projects.
- jCORA does not support the revise phase at all.

The two comments above indicate a considerable need for further development of the CBR tool jCORA.

3.3.1.2 Using the CBR tool jCORA for case specification

We explain the functionality of the CBR tool jCORA below using screenshots as an example. Since jCORA, as a tool for ontology-supported *case*-based reasoning, focuses on the use of the term “Fall” for “*case*”, the following will primarily refer to “Fälle”— instead of the underlying and synonymous *projects*. Nevertheless, the content of each case refers to the specification of a project about which knowledge—primarily of the natural-language, experience-based type—is to be stored in the project knowledge base of an ontology-supported CBR system and reused for the planning, management, and control of new projects.

To create a new case at individual level in the CBR tool in jCORA, click on the “+” symbol on the jCORA start page:

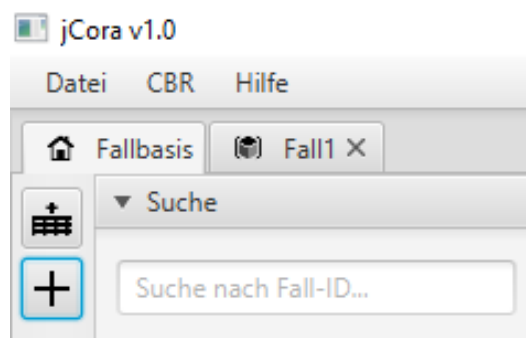


Figure 69: Creating a new case

A case ID can then be selected for the case. This step is demonstrated in Figure 70 below, with the designation “Kooperative_Leitstelle” selected for the case.

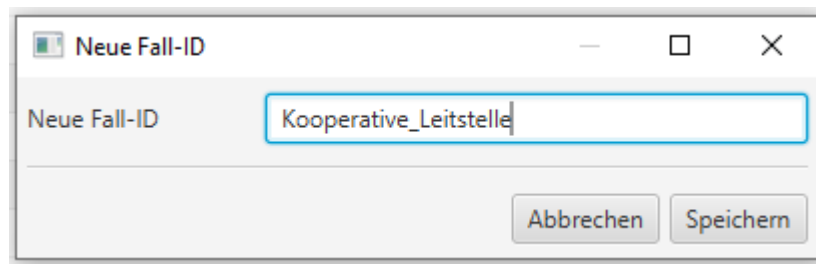


Figure 70: Creating a case ID

As soon as the case has been created, the three-part case structure already shown is automatically constructed and displayed; cf. Figure 71 below.

A *case graph* is used to represent each case. Each of its *nodes* initially lists an individual in bold in its first (top) line, which is used to specify a case at the *individual* level. In addition, the second (bottom) line of the same node shows, in normal print, the *class* to which the node's individual belongs in the underlying ontology. In Figure 71 below, for example, the individual *Fall* belongs to the class *Projekt*, just as the individual *Fall Beschreibung* belongs to the class *Projektbeschreibung*.

The directed *nodes* between the nodes of a case graph each represent a *non-taxonomic relation*, each of which is identified as an edge label, with the following applying: The two *individuals* from the pre-area of the relation (labeling of the node at the start of the edge in the first line) and the post-area of the relation (labeling of the node at the end of the edge in the first line) represent a *relation element* of the non-taxonomic relation as an ordered pair, which is identified as an edge label. In addition, due to the double labeling of the nodes of the case graph, each directed edge between two nodes of a case graph is labeled with the non-taxonomic *relation* that links the two classes from the domain of the relation (labeling of the node at the start of the edge in the second line) and the range of the relation (labeling of the node at the end of the edge in the second line).

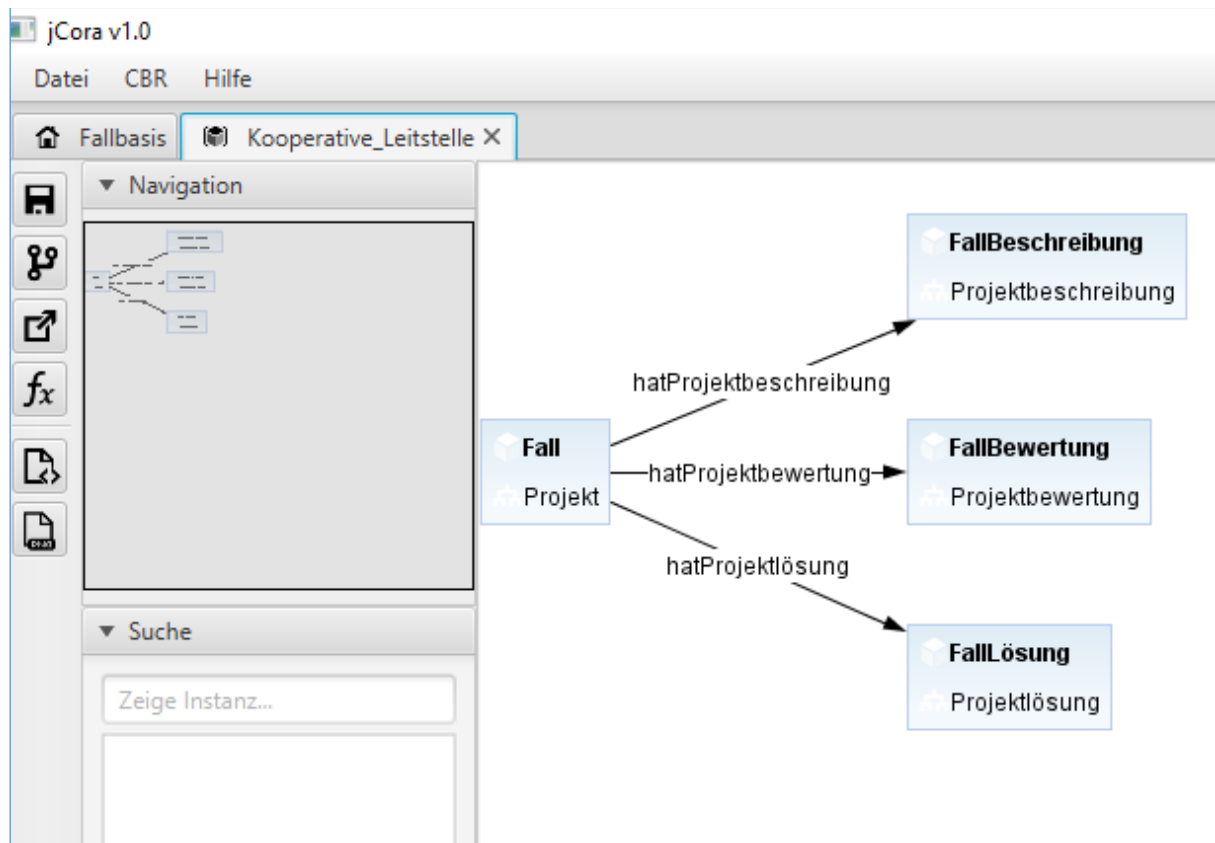


Figure 71: Original case graph in jCORA

The individual *Fall* is a special feature in jCORA. Since it was neither defined as a global individual in Protégé nor is available as a conventional, case-specific individual in jCORA, it cannot be classified in the categorization of global and local individuals. Instead, this individual is defined by jCORA's settings, as shown in Figure 72 below. Semantically, it makes no sense to consider an individual *Fall* as a subclass of *Projekt*, as a case per se is not a subclass of a project—rather, the terms “Projekt” and “Fall” are used synonymously. This argumentation also applies analogously to the individuals *FallBeschreibung*, *FallBewertung*, and *FallLösung*.

The screenshot shows the 'Einstellungen' (Settings) window in jCORA, specifically the 'Fallstruktur' tab. The window has three tabs: 'Sprache', 'Fallbasis', and 'Fallstruktur'. The 'Fallstruktur' tab is active. It contains three sections of settings, each with a label on the left and an input field on the right.

Section	Label	Value
Konzept	"Fall"	Projekt
	"Fallbeschreibung"	Projektbeschreibung
	"Falllösung"	Projektlösung
	"Fallbewertung"	Projektbewertung
Relation	"hatFallbeschreibung"	hatProjektbeschreibung
	"hatFalllösung"	hatProjektlösung
	"hatFallbewertung"	hatProjektbewertung
Instanzname	"Fall"	Fall
	"Fallbeschreibung"	FallBeschreibung
	"Falllösung"	FallLösung
	"Fallbewertung"	FallBewertung

At the bottom right of the window, there are three buttons: 'Zurücksetzen', 'Abbrechen', and 'Speichern'.

Figure 72: Settings in jCORA

To specify a safety-critical IT project, a new non-taxonomic relation is selected by right-clicking on the individual *FallBeschreibung*. Figure 73 below illustrates the selection of possible non-taxonomic relations, starting from the individual *FallBeschreibung*. Only those non-taxonomic relations are displayed that have the class *Projektbeschreibung* in their respective pre-fields.

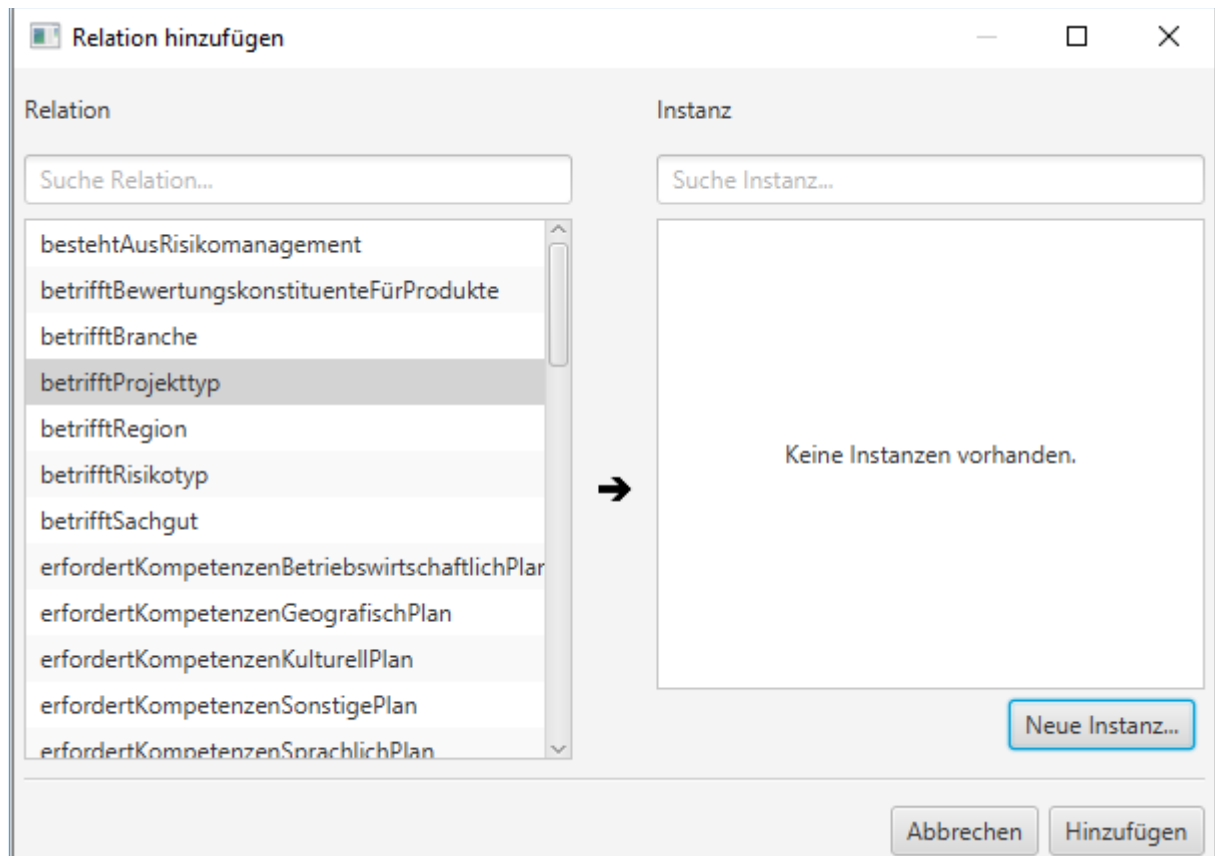
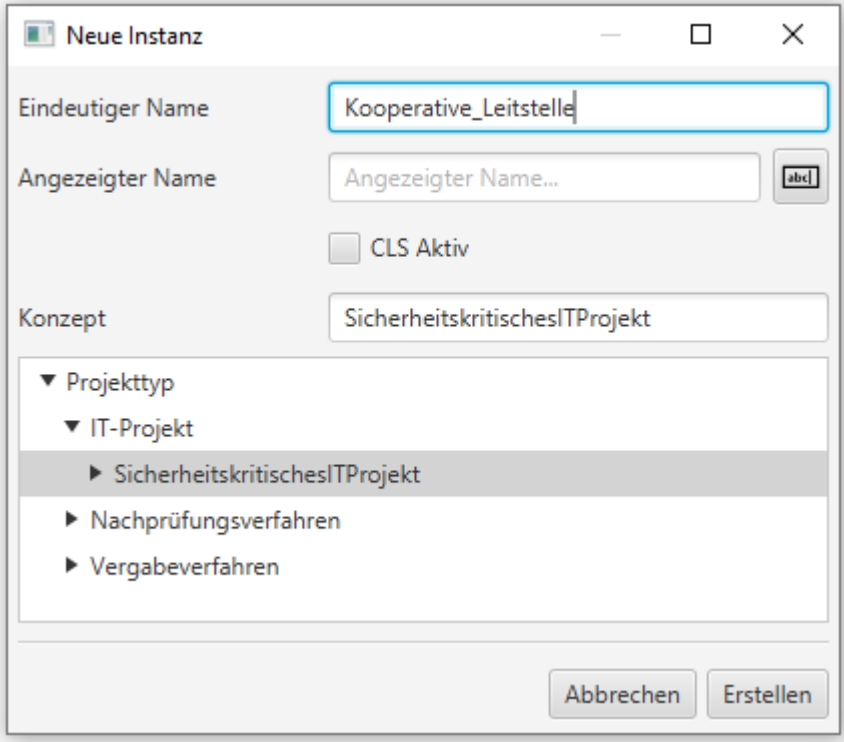


Figure73: Possible non-taxonomic relations based on the individual *FallBeschreibung*

In the present context, the non-taxonomic relation *betrifftProjekttyp* is selected in order to assign an individual that belongs to the class *SicherheitskritischesITProjekt*, as this is required in the descendant area of the non-taxonomic relation *betrifftProjekttyp*. As no individuals of the class *SicherheitskritischesITProjekt* exist yet, a new individual called *Kooperative_Leitstelle* is created and assigned to the class *SicherheitskritischesITProjekt*, as illustrated in Figure 74 below.



Neue Instanz

Eindeutiger Name: Kooperative_Leitstelle

Angezeigter Name: Angezeigter Name...

☐ CLS Aktiv

Konzept: SicherheitskritischesITProjekt

- ▼ Projekttyp
 - ▼ IT-Projekt
 - ▶ SicherheitskritischesITProjekt
 - ▶ Nachprüfungsverfahren
 - ▶ Vergabeverfahren

Figure 74: Creating the new individual *Kooperative_Leitstelle*

After constructing the individual *Kooperative_Leitstelle*, it can be used in the sub-area of the non-taxonomic relation *betrifftProjekttyp*, as shown in Figure 75 below.

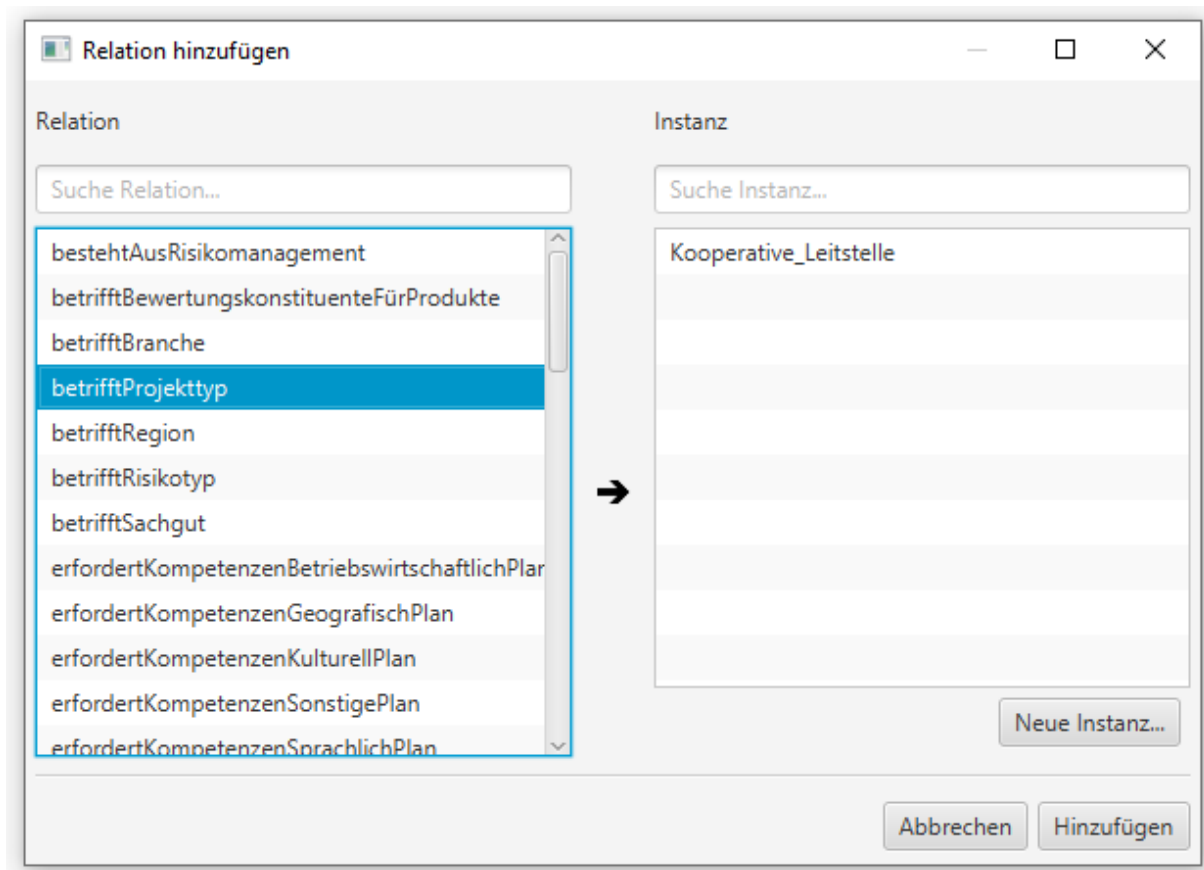


Figure 75: Selection of the individual *Kooperative_Leitstelle* as an element of the sub-area of the non-taxonomic relation *betrifftProjekttyp*

The added relation element of the non-taxonomic relation *betrifftProjekttyp* is shown in the display window as an extension of the case graph by the individual *Kooperative_Leitstelle* from the class *SicherheitskritischesITProjekt*.

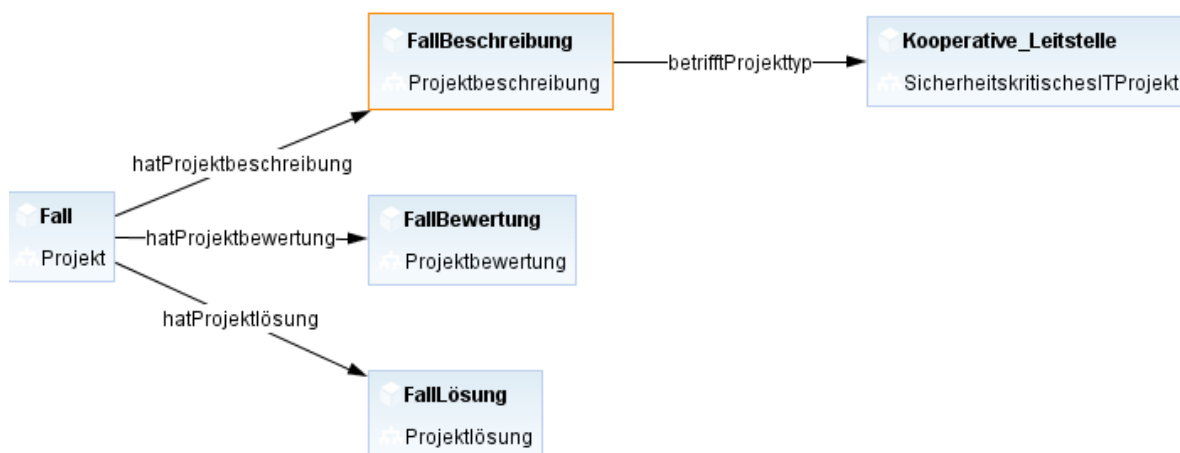


Figure 76: Extended case graph in jCORA

In the following, selected properties of a safety-critical IT project are assigned to the individual *Kooperative_Leitstelle* as an example. Figure 77 below shows an example of how the case graph can be extended by non-taxonomic relations based on the individual *Kooperative_Leitstelle*. The display window shows only those non-taxonomic relations to which the class *SicherheitskritischesITProjekt* is assigned in the domain.

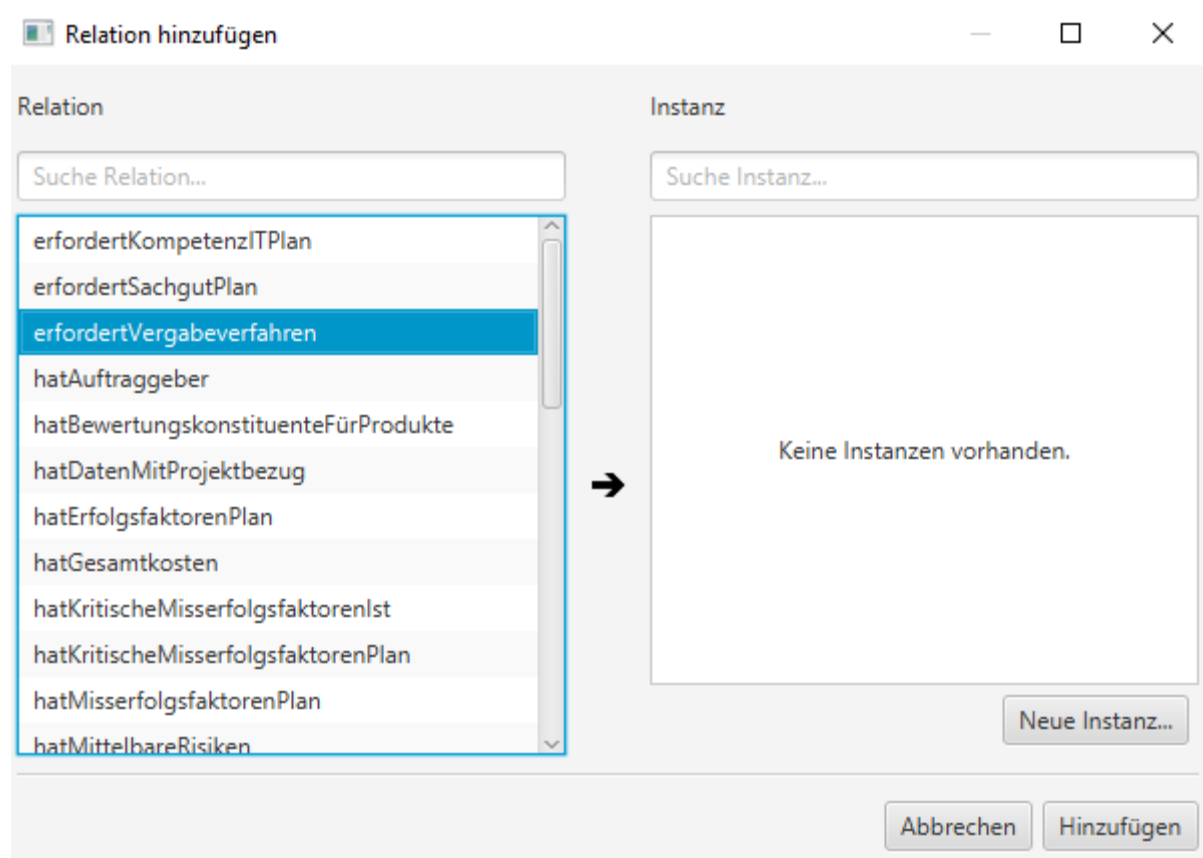


Figure 77: Display window for adding a non-taxonomic relation

Figure 78 below illustrates the assignment of attribute values for the attributes of the individual *Kooperative_Leitstelle*. The “+” symbol is selected to assign specific values to the attributes. The CBR tool jCORA automatically assigns the selected attribute the data type that was defined for the respective attribute in the underlying ontology.

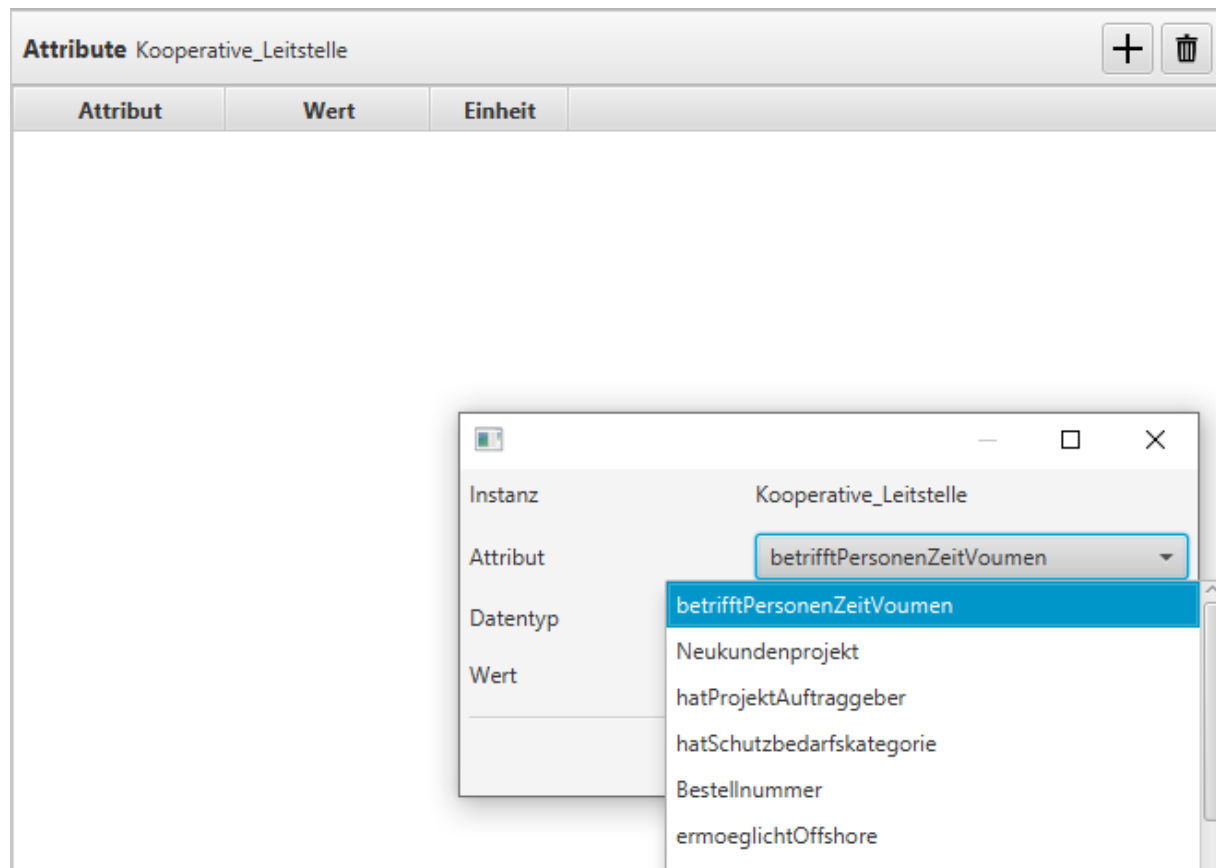


Figure 78: Assignment of values for the attributes of the individual *Kooperative_Leitstelle*

Figure 79 below illustrates the assignment of values for its attributes for the individual *Kooperative_Leitstelle* as an example.

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)	

Figure 79: Assigned values for attributes of the individual *Kooperative_Leitstelle*

In this way, the user can extend the case graph of jCORA with individuals and associated relation elements and attribute values, until they have created all knowledge components that are considered relevant for the description of the safety-critical IT project under consideration.

[illegible]

Figure 80 shows an example of the knowledge components associated with the case description. As required in the definition of a safety-critical IT project in this article, the safety-critical IT project “ZentraleLeitstellePolizei” has already undergone an award procedure. Its suitability criteria included the project reference and the required turnover

in the tendered area. As part of this award procedure, which was a negotiated one, the fictitious company ITBeratungAG was commissioned. The contract award procedure includes a requirement for geodata maintenance in the service description. Said maintenance is solved in the security-critical IT system by software (ArcGIS for Server Advance), hardware (Bull Sequana), and data (Naveteq and OSM data), which represent both tangible and intangible real assets. The hardware has the IT security features of hardening and geo-redundancy. This requirement is implemented in the production system, called PU1. One of the customer's project employees, Police Chief Inspector (PHK) Max Mustermann, who works at Haltern Police Headquarters and is part of the project, has expert power over geodata maintenance and can therefore influence the respective requirement. In a brainstorming session, it was determined that this requirement carries the risk that the process of geodata maintenance is not defined and therefore the requirement is difficult to implement.

Although this situation lays out a fictitious case and fictitious requirements, it is representative of a complex safety-critical IT project. The light blue nodes of the case graph from jCORA represent local individuals. The orange nodes, on the other hand, represent global individuals that were already created during ontology construction using Protégé, such as procurement steps and risk management measures.

3.3.1.3 Use of the CBR tool jCORA for case-related similarity calculation

Once the user has completed a case description—i.e., a new project's description—in the CBR tool jCORA, a *similarity calculation* is performed between the new case (synonym: project) and the old cases (synonym: projects), about which project management-relevant knowledge, in particular natural language experience knowledge, is stored in the CBR tool's project knowledge base. The similarity calculation between cases is carried out in the CBR tool jCORA by calling the function “Create a CBR request from this case”, as illustrated in Figure 81 below.

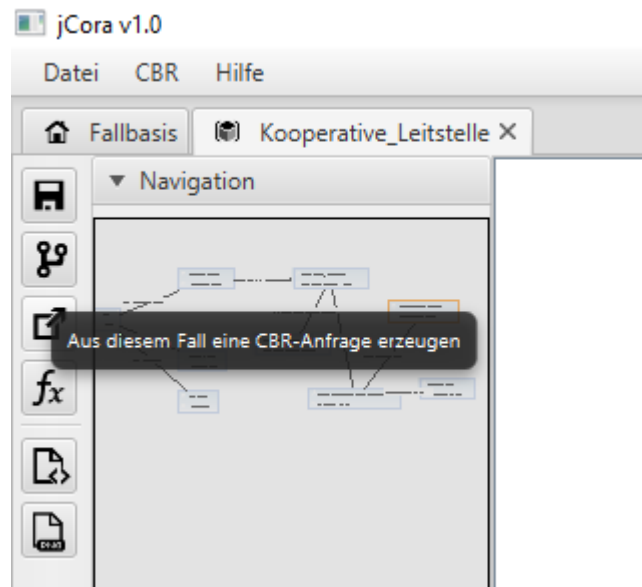


Figure 81: Create CBR request in jCORA

The similarity calculation takes place in a new window, in which the relations and attributes as properties of a case (project) can be weighted according to the user's own preferences with regard to their relevance for the similarity calculation using percentage values, as illustrated in Figure 82 below. In addition, the CBR tool jCORA offers the option of saving user-specific weighting profiles and calling them up again for later use.

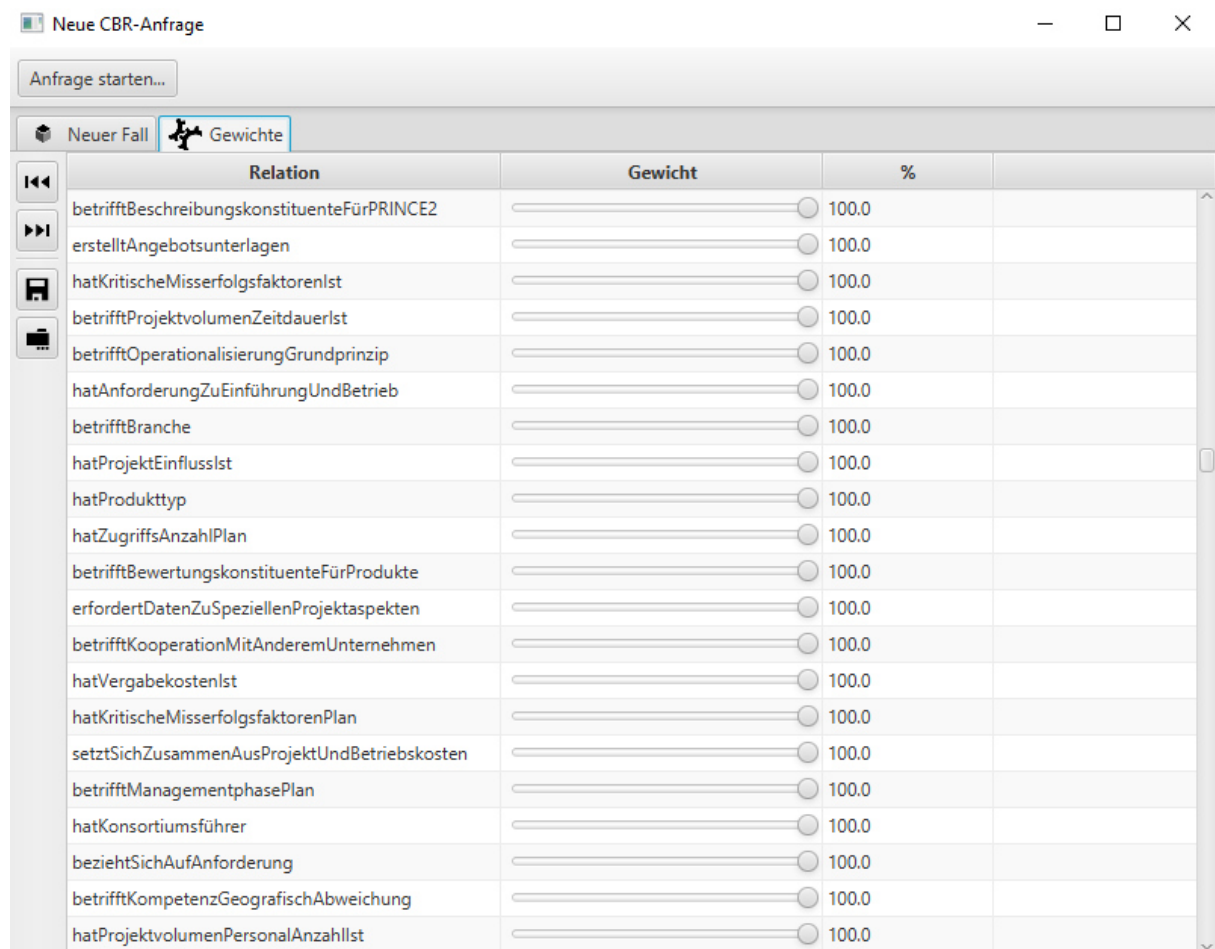


Figure 82: Setting the weighting of the properties

Once the weighting has been defined, the user can start the similarity calculation. This is done by calling up the “Anfrage starten” function. The similarity calculation is performed for all cases that exist in the CBR tool’s case base, i.e., its project knowledge base. Figure 83 below illustrates the presentation of results after the similarity calculation. It should be noted that the graph presentation in Figure 83 can be considered inadequate, as it lacks labeled values along the x and y axes and it therefore remains unclear which specific results are shown. A simple loading bar would enhance the user’s experience, as the results of the similarity calculation are already displayed in a separate window.

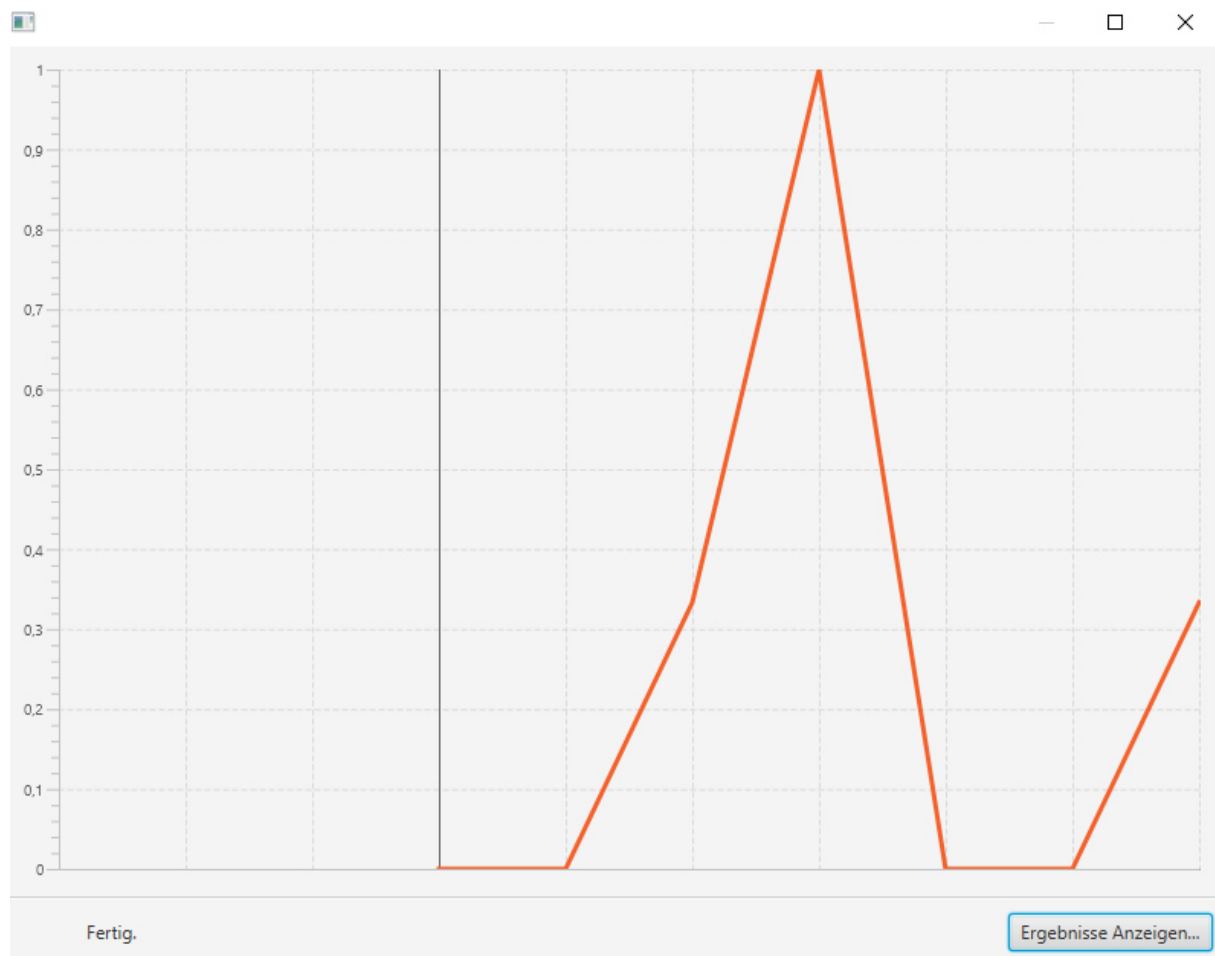


Figure 83: Graphical presentation of results in jCORA

Once the similarity calculation has been completed, the user can call up the “Show results” function, which displays the results of the similarity calculation in the following format:

Fall-ID	Ähnlichkeit		Adaptieren	Anzeigen
Aufbau_Kooperative_Leitstelle	100%		<button>Adaptieren</button>	<button>Anzeigen</button>
Neuausrichtung_eines_Einsatzf_hrungssy...	36%		<button>Adaptieren</button>	<button>Anzeigen</button>
Aufbau einer zentralen Datenbank	26%		<button>Adaptieren</button>	<button>Anzeigen</button>

Figure 84: Presentation of results in jCORA

The calculated similarity between the projects is shown as a percentage and sorted from highest similarity to lowest similarity. In addition, the CBR tool jCO-RA offers a filter function to display only those similar projects that achieve at least a certain similarity

value. The user could, for example, specify a minimum similarity of 75% in the filter function.

3.3.1.4 Limitations of the CBR tool jCORA

We have already mentioned some of the CBR tool jCORA's limitations (restrictions) for use in an operational environment; cf. also WEBER et al. (2023), pp. 24–25; HERDER/ZELEWSKI/SCHAGEN (2022), pp. 44–45; ZELEWSKI/SCHAGEN (2022), pp. 51–54; BERGENRODT/KOWALSKI/ZELEWSKI (2015), pp. 537–541. We here describe further fundamental limitations to which the CBR tool jCORA is subject with regard to its use in an operational environment.

The most important of these, which have already been discussed in other publications (particularly those mentioned above), are briefly outlined below.

Similarity algorithm: jCORA's similarity algorithm leads to incorrect similarity calculations if global individuals are used incorrectly. Such improper use occurs when global individuals—contrary to their definition as “globally valid”—are not used unchanged (“constant”) for all projects, but are modified within the specification of individual projects, e.g., with regard to the individual values. In this respect, there is no error in the similarity algorithm; rather, the problem is an impermissible algorithm application. However, jCORA users perceive the resulting, seemingly implausible calculation results as an algorithmic error. Adding further specific similarity functions is only possible through implementations in the jCORA source code. Only IT specialists very familiar with jCORA can accomplish this in an error-free manner. This proves to be a considerable limitation on jCORA's use in operational practice, which often requires flexible adaptations of specific similarity functions to the requirements in their project management environment. Furthermore, the current lack of a generally recognized algorithm for calculating the similarity between projects on the basis of project ontologies proves to be unsatisfactory.

We will briefly digress to illustrate the previously mentioned “error-proneness” of jCORA's similarity algorithm when using global individuals. In the safety-critical IT project ontology, we constructed some global individuals (e.g., the awarding steps as well as the PRINCE2 project management method), which, however, lead to incorrect similarity values when used in jCORA, as already noted by WEBER et al. (2021), p. 26. Accordingly, in the following three cases we used a global individual only for “realignment of an operational management system”, in order to avoid incorrect values in the similarity calculation.

Although global individuals can be created in an ontology editor such as Protégé, they can lead to errors in jCORA's similarity calculation if they are modified "locally" (i.e., individual-related) during the specification of project individuals ("cases") by adding new attributes or relations. This could lead to a contradiction between the global individual definition in Protégé on the one hand and the local individual modification in jCORA on the other. Under this premise, global individuals would not be error-prone per se, as long as users ensure they never "modify" them in jCORA in the aforementioned way. However, it is equally legitimate to criticize that not modifying global individuals potentially results in a loss of available or constructed linguistic means of expression.

Figure 85 below illustrates the problem when the similarity to other cases and to itself is calculated based on the case "realignment of an operational command system". jCORA gives said similarity to itself as 98%, which is not correct.

Fall-ID	Ähnlichkeit		Adaptieren	Anzeigen
Neuausrichtung_eines_Einsatzf_hrungssystems_der_Polizei	98%		Adaptieren	Anzeigen
Aufbau_Kooperative_Leitstelle	36%		Adaptieren	Anzeigen

Figure 85: Incorrect result of the similarity calculation when using global individuals

Finally, the similarity value calculated in jCORA must also be critically evaluated. In this article, we calculated the similarity between the three cases in chapter 3.3.3.2. We based this calculation on the so-called universal similarity function used by default in jCORA. To overcome this limitation, later in this contribution we implement a specific similarity function that can calculate the similarity between words using the Word2Vec technique. The implementation as a serverless function makes it possible to add further specific similarity functions without having to change the existing source code of an existing serverless function.

Similarity tables can be loaded, but a specific similarity function has to be implemented in the source code, which is not trivial in a monolithic application, as it is a source code block. In addition, the source code is not documented in a technical manual. Although the source code is well documented, an architectural plan of the CBR tool jCORA would be desirable in order to better understand the relationships between the tool's classes.

Furthermore, the universal similarity function $sim_{numeric}$ must be critically scrutinized with regard to its plausibility, as the values in the ontology vary greatly depending on the context (e.g., *TCVBC* or *hatAlter*) and the similarity is calculated on the basis of the largest and smallest value. Even with the aforementioned limitations in the case specification, which can be understood as additional limitations of jCORA, the results in

chapter 3.3.3.2 show that a tendency of similarity between the cases can be calculated in order to gain useful insights for the reuse of experience-based knowledge from safety-critical IT projects. Our following explanations of the cases do not cover all areas of a case, as that would make them disproportionately long. Instead, we explain selected areas, such as the requirements of the service descriptions that serve as the basis for the real assets used.

Expressive power of the underlying ontology: The performance of jCORA depends largely on the underlying ontology, in particular on its expressive power. Restrictions in the ontology, for example due to limitations of the ontology editor such as Protégé, can have a negative effect on said performance. This includes, for example, the fact that—as already mentioned—only two-digit non-taxonomic relations can be used in Protégé and, based on this, also in jCORA. Yet business practice sometimes also requires at least three-digit (or even “higher”) non-taxonomic relations as linguistic means of expression for a “natural” modeling of the relevant business facts. This applies, for example, to non-taxonomic relations for competence relations, which should include at least one competence holder (actor), one competence type, and the extent of competence fulfillment as relation points. This could be supplemented, for example, by information on proof of possession of a competence (certificates) and information on the period during which the competence was acquired—or possibly forgotten due to non-use.

Visualization of instantiated project ontologies using case graphs: The limitations that can be identified with regard to visualization are reflected in the CBR tool jCORA’s fundamentally poor usability. The usability limitations for this software’s use in an operational environment are described in more detail by WEBER et al. (2023), pp. 53–56.

Help tools for jCORA: A number of publications on jCORA now exist. However, these are not an adequate substitute for user-related online help, which would significantly support the tool’s use in an operational environment. In this respect, the publication SCHAGEN et al. (2022) is worth mentioning; it presents the development of an e-learning module for user-friendly familiarization with the use of jCORA.

Adaptation: The adaptation of solutions for very similar old projects (cases) to the descriptions of new projects (cases) represents a problem that has not yet been satisfactorily solved, not only for the CBR tool jCORA in particular, but also for CBR systems in general, for the promising use of case-based reasoning in operational practice with regard to the reuse of experience-based knowledge in project management. We will discuss this in more detail shortly, with reference to jCORA’s special limitations.

User interface: The limitations regarding jCORA's usability are discussed in more detail by WEBER et al. (2023). They report on usability tests carried out with project managers from the area of safety-critical IT projects, among others. Overall, they did not give jCORA a convincing testimonial; cf. WEBER et al. (2023), pp. 39–56.

For clarification purposes, we discuss only the limitation of adaptation in more detail below, as the adaptation of solutions for similar old projects to the descriptions of new projects plays a vital role in case-based reasoning. In the sense of the CBR cycle's reuse phase presented earlier, an old project and its knowledge components (project solution and project evaluation) can be reused by calling the “Adapt” function. An adaptation rule is selected in the “Existing rules” window and executed using the “Apply” function. This applies the selected adaptation rule to the solution for a new project.

Figure 86 below illustrates that the CBR tool jCORA currently only has one adaptation rule: “Copy solution”. This adaptation rule is used to copy the entire solution of an old project into the solution for a new project. This copy function adopts the project solution for the most similar old project as a proposed solution for a new project without making any adjustments. The necessary adaptation of the project solution for a most similar old project to the new project under consideration must therefore be carried out “manually”. This adaptation rule is also referred to as “zero adaptation”; cf. WILKE/BERGMANN (1998), p. 500.

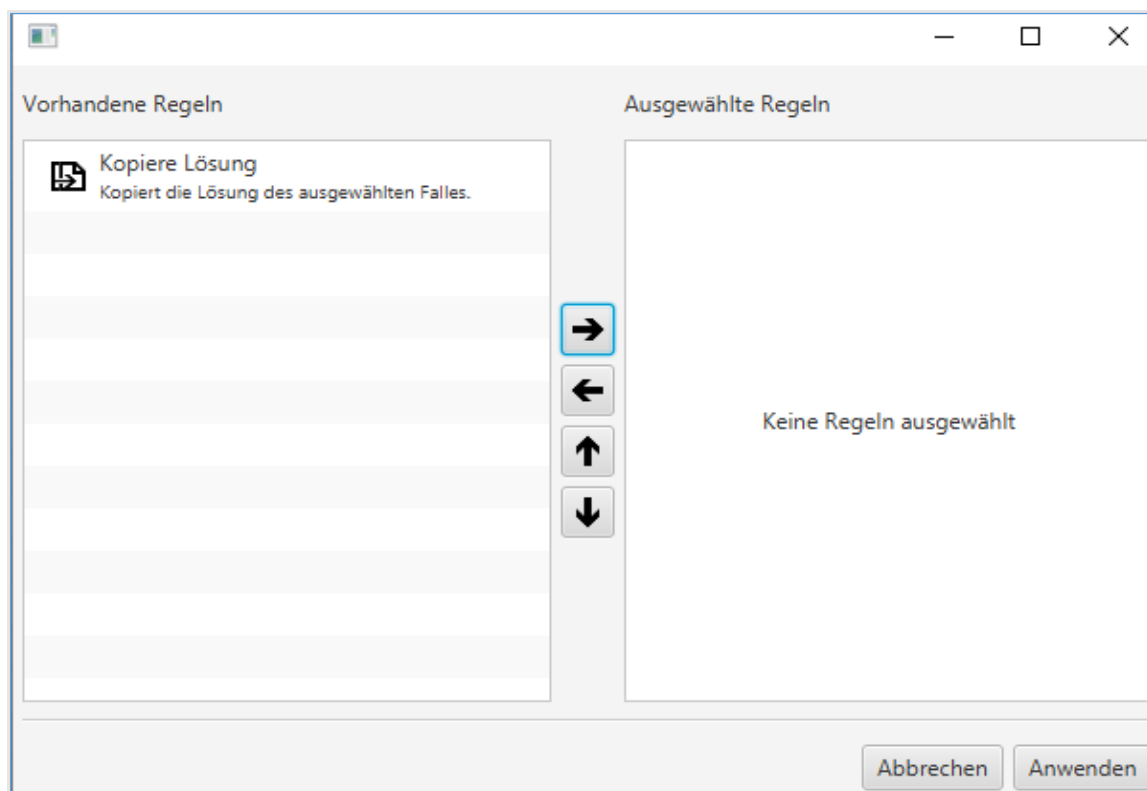


Figure 86: Adaptation window in jCORA

Zero adaptation is highly controversial because copying the solution for an old project as a proposed solution for a new project contradicts a project's "uniqueness", which is often emphasized in the definition. Instead, experts sometimes suggest using the solution for an old project as a starting solution, in order to arrive at a solution for the new project under consideration by means of "manual" adjustments. However, this "manual" adaptation of solutions for safety-critical projects contradicts the requirement to be able to manage new safety-critical IT projects as far as possible with "intelligent", AI-based knowledge management systems with *computer support*.

However, the automatic adaptation of solutions from old to new projects in a CBR tool is a *field of research* that still requires extensive and in-depth further investigation. Currently, it is suffering from far too many research and implementation gaps in the state of the art of CBR tools.

The CBR tool jCORA already offers a generic adaptation function—"adapt"—which can be used to define user-specific adaptation rules; cf. ZELEWSKI/SCHAGEN (2022), pp. 40–41; BERGENRODT/KOWALSKI/ZELEWSKI (2015), pp. 534–536. However, this generic adaptation function is still not sufficient because it does not define user-, project-, or industry-specific adaptation rules. In this respect, the CBR tool jCORA—like other CBR tools—has a serious application gap.

In the following, we discuss in more detail a particular limitation of the CBR tool jCORA that the relevant specialist literature (with the exception of SETHUPATHY (2024), pp. 341–346) has far neglected to deal with—namely, jCORA's monolithic application structure. It represents a central weakness for jCORA's practicability in an operational environment. We explain it in more detail here, and present a comprehensive, cloud-based approach to overcoming this vulnerability later, in chapter 4.

The central importance of jCORA's monolithic application structure rests on the fact that current technological developments in the cloud area are increasingly replacing monolithic applications in the operational environment; cf. FRANK/SCHUMACHER/TAMM (2019), p. 6; FRITZSCH et al. (2019), pp. 128–129. The disadvantages of a monolithic application in an operational environment are described in detail by OLIVEIRA ROCHA (2021), pp. 4–9, among others. They support the assumption that current technological developments in the cloud area will largely replace monolithic applications in the operational environment.

Although the CBR tool jCORA's limitation with regard to its monolithic application structure represents a serious obstacle to its use in an operational environment, it has not yet been addressed in any scientific publication (with the exception of SETHUPATHY (2024); see above). HERDER/ZELEWSKI/SCHAGEN (2022), pp. 44–45, criticize jCORA's

failure to offer any “practical” interfaces. However, they do not address the underlying limitation due to the monolithic application structure. This does not take into account the fact that, by breaking up a monolithic application structure, new types of interfaces can be offered that can be used by other application software, such as Microsoft Office products, which are particularly widespread in business practice. We explain this in detail below. However, it should be noted in advance that a separate serverless function must be implemented for each new type of interface. This means that the resources required for cloud-based reimplementation of the CBR tool jCORA should not be underestimated. However, it also promises great additional benefits in terms of greater application flexibility and new interfaces to widely established application software.

The CBR tool jCORA has a monolithic application structure, in which the functionalities are bundled in a single application, and is to be classified in the 1-tier architecture, so that no data storage level exists. For example, a data storage tier could have been implemented using a common database to separate the presentation tier and the data storage tier (2-tier architecture). In principle, Protégé supports the use of a database as a backend component, so that any application could access the database and the ontologies constructed in Protégé could be called up. The support of databases in Protégé as well as the necessary connection formalities to a database (connection strings) are described in PROTÉGÉ (2010). In jCORA, it would have been conceivable to store the underlying ontology and the project knowledge base in a standard database and integrate it into jCORA using interfaces. Using database triggers, the necessary formalities—such as the existence of the case structure (project structure) with case description, case solution, and case evaluation—could have been checked automatically. In addition to Java technology—used for jCORA—database technologies such as SQL could have been implemented to construct automated check scenarios in the database (project knowledge base), for example. However, the restriction of a monolithic application structure would still be present if this separation existed. As this fundamental separation is missing in jCORA, its monolithic application structure represents a fundamental limitation of the CBR tool’s use in an operational environment.

The fundamental separation between the presentation layer and the data storage layer would support jCORA’s maintainability, as the coupling between the layers is kept to a minimum and clear interfaces exist between the two layers; cf. RAU (2016), p. 304. This subdivision would also support portability, making it easier to replace the database or use a different presentation level without changing the data storage level.

Even if jCORA were to support this fundamental separation between the data storage level and the presentation level, the CBR tool would still be a monolithic application from an implementation perspective. In jCORA, the cases of the case base (projects of

the project knowledge base) are stored in a proprietary .dat and .idn file. Figure 87 below illustrates the .dat and .idn files that are “automatically” created in jCORA’s “Case base” folder.

Fallbasis		Fallbasis durchsuchen	
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

Figure 87: .dat and .idn files from jCORA in the *case base* folder

A .dat file is a generic data file created by a specific software. In most cases, the associated software is the only way to open this data file, which in turn reinforces the monolithic application structure. The .idn file extension is not generally defined. Due to a lack of technical documentation for the CBR tool jCORA, the purpose for which these .idn files are created remains unclear. However, it has been observed that .idn and .dat files are automatically created in the “Case base” folder when cases are constructed.

In addition to the fundamental restriction imposed by the 1-tier architecture, monolithic CBR tools (“applications”) such as jCORA have other general disadvantages when used in an operational environment, which are briefly summarized in the following points:

- **Technology:** When adapting a monolithic application structure, the entire affected software must be tested for functionality. This means that software development becomes more complex and slower; cf. FRITZSCH et al. (2019), p. 129; RAU (2016), p. 304. Furthermore, changes in a monolithic application structure impact the entire application, which leads to time-consuming adjustments and thus increases costs; cf. FRANK/SCHUMACHER/TAMM (2019), pp. 161–162. The high level of complexity and the many dependencies within the monolithic application structure make it difficult to localize a possible error in the software; cf. OLIVEIRA ROCHA (2021), pp. 6 and 13. Test procedures must be more complex in order to rule out any errors that may arise from a change to the software due to the internal dependencies. A monolithic application structure is usually written exclusively in one programming language. Technological advances, such as those resulting from new programming languages, cannot be taken into account; cf. OLIVEIRA ROCHA (2021), p. 15; FRITZSCH et al. (2019), p. 129.
- **Deployment:** Any changes to software with a monolithic application structure require a complete deployment of said software. With jCORa, the software would have to be deployed to all affected workstations by means of software distribution or on-site installation. Software distribution tools (application release automation) are automation tools for distributing software to different target systems; cf. PFITZINGER/JESTÄDT (2017), p. 582. Both variants would incur costs for deployment and downtime during deployment. Furthermore, it may be necessary to adapt the affected workstations. With the Java-based CBR tool jCORa, for example, it may be necessary to update the Java runtime or the environment variable, which in turn requires additional effort and represents a further source of errors. An environment variable (path parameter) designates configurable paths in an operating system to certain software. The disadvantage of defining an environment variable is that it must always be adapted via a client's system control. In the case of a distributed client environment across several locations, this would mean increased effort, as an adjustment would have to be made in each client environment.
- **Mobility:** At present, the CBR tool jCORa cannot be run on mobile devices. This corresponds to the usual pattern for monolithic applications, as they are usually developed for desktop clients. Therefore, the development of a separate software variant would be necessary to enable its mobile executability. The use of software on mobile devices is an important requirement with regard to its use in operational practice. This is justified by the fact that the proportion of mobile internet users in Germany was 82% in 2021; cf. INITIATIVE D21 (2022), pp. 14–15. The

proportion of mobile device owners in the German population was around 88.8% in 2021; cf. TENZER (2022). The figure relates to people aged 14 and over who have a smartphone or cell phone in their household. The CBR tool jCORA, which is Java-based software, cannot be used on mobile devices with the operating system (iOS) by default. This is because the iOS operating system does not support the Java programming language, only its own Swift programming language. It could conceivably be used on mobile Android devices, as Android supports the Java programming language as a standard programming language; cf. MAWLOOD-YUNIS (2022), p. 55. However, technologies exist to equally support both mobile device types, which represent more than 99% of the market share; cf. KANTARWORLDPANEL (2022).

- Scalability: Individual components of the CBR tool jCORA cannot be scaled. Scaling is usually achieved by duplicating the entire software, for example on another—possibly only virtual—computer; cf. OLIVEIRA ROCHA (2021), p. 15; FRITZSCH et al. (2019), p. 129. However, this scaling approach is an inefficient way to respond to rapidly changing workloads while remaining optimal in terms of resource utilization; cf. FRITZSCH et al. (2019), p. 129. One criterion for evaluating inefficiency is the relationship between the hardware resources used and computing times. An optimality criterion in this context can be the achievement of maximum performance with the lowest possible latency. This criterion aims to achieve the highest performance without compromising latency. This means that a large number of requests can be processed quickly without unnecessarily slowing down the response time (latency).

Software with a monolithic application structure can have the advantage, at least in the early phases of development, of developing the software quickly and reducing the cognitive effort for code management and deployment; cf. HARRIS (2022); OLIVEIRA ROCHA (2021), pp. 10–11. Particularly in the area of rapid prototyping, software with a monolithic application structure could at least help to quickly develop an executable prototype. By using only one code base (in this case the Java programming language), the entire development can take place in one development environment, e.g., Eclipse. This could simplify software development, especially at the beginning of the process; cf. HARRIS (2022). Further advantages of monolithic application structures are cited in HARRIS (2022) and OLIVEIRA ROCHA (2021), pp. 9–13. Overall, however, current publications come to the conclusion that the disadvantages of software with monolithic application structures for use in an operational environment outweigh the advantages; cf., e.g., OLIVEIRA ROCHA (2021), p. 13; FRANK/SCHUMACHER/TAMM (2019), pp. 153–154.

For company-relevant software, monolithic application structures also represent an organizational bottleneck in particular; cf. FRANK/SCHUMACHER/TAMM (2019), pp. 153–154. In their opinion, a software's performance essentially determines the performance of the business model based on it. For an ontology-supported CBR system, such as the CBR tool jCORA considered here, which is used for the reuse of experience-based knowledge in the operational environment of project management, a monolithic application structure represents a considerable business risk due to the aforementioned difficulties (limitations).

Ideally, software with a monolithic application structure works without errors. If an error does occur, however—for example in one of the CBR cycle's four phases—or if a sub-component of the monolithic application structure—for example the case base (project knowledge base) of the CBR tool jCORA—is overloaded, it is very likely that the entire software will produce an error case and can therefore no longer be used. The availability of the entire software is therefore at risk. Its use as a central knowledge management system could lead to a considerable operational risk in the event of a “total failure”.

The CBR tool jCORA's monolithic application structure makes its flexible further development and the rapid provision of additional functionalities considerably more difficult. Such further development may even be fundamentally thwarted due to “prohibitively” high costs. Therefore, a non-monolithic application structure in which individual functions (e.g., for individual phases of the CBR cycle, such as the currently not yet satisfactorily covered reuse phase) could be developed and provided independently of other components (such as the user interface) would be much more flexible. One possible solution to the problem outlined above is so-called serverless functions, which we explain in detail later in this article.

In addition to the aforementioned disadvantages of software with monolithic application structures, such “monolithic” software no longer corresponds to the current state of “modern” software technology, particularly with regard to economic aspects; cf. FRANK/SCHUMACHER/TAMM (2019), p. 167. Infrastructure automation for the use of software in the operational environment has developed considerably in recent years; cf. FOWLER/LEWIS (2015), p. 19. Technological developments in cloud computing—in particular the offerings of “hyperscalers” such as Amazon Web Services (AWS) and Microsoft Azure—have significantly reduced the operational complexity of developing, deploying, and operating software with non-monolithic application structures (microservices) and have overtaken the use of software with monolithic application structures in the operational environment in terms of the “state of the art”; cf. JAMSHIDI et al. (2018), pp. 26–27; FOWLER/LEWIS (2015), p. 5. In this context, software used in operational practice is

increasingly developing into cloud-native applications that are provided and further developed exclusively in the cloud. We explain the development of cloud-based, ontology-supported, case-based reasoning as a special use case of such cloud-native applications in detail in chapter 4.

3.3.2 Description of the cases to represent three practical examples

3.3.2.1 Preliminary remarks on the three practical examples

We then constructed three practical examples using the CBR tool jCORA and later analyzed their similarities. On the one hand, these practical examples are intended to illustrate how projects can be represented as “cases” in an ontology-supported CBR system and how the experience-based knowledge from old, already completed projects can be reused for the planning, implementation, and control of new projects using a CBR tool such as jCORA. On the other hand, they illustrate how safety-critical IT projects can be specifically recorded within an ontology-supported CBR system such as the CBR tool jCORA.

The following three cases—in which “practical examples” and “safety-critical IT projects” are synonymous—do not represent real-life, safety-critical IT projects. However, they help to identify the design of a safety-critical IT project that has a previous award procedure and fulfills requirements for a safety-critical IT system that can be classified in the aforementioned requirement areas. We have deliberately based the case designations on common projects. Table 54 below lists the three examined cases (projects).

Case No.	Case Name
1	Neuausrichtung eines Einsatzführungssystems der Polizei
2	Aufbau einer kooperativen Leitstelle
3	Aufbau einer zentralen Datenbank für Ermittlungen

Table 54: Practical examples of security-critical IT projects

In addition to the fundamental limitations of the CBR tool jCORA mentioned in chapter 3.3.1.4, we must address further restrictions that have become apparent with regard to the three practical examples (“cases”) of safety-critical IT projects considered in the following from a *case-specific* perspective.

Strictly speaking, these three cases do not represent complete cases, but merely include some requirements from safety-critical IT projects. We were forced to make this restriction because the CBR tool jCORa sometimes runs into an error with a more comprehensive case construction and subsequent similarity calculation if too many or too extensive cases (individuals) are considered; see Figure 88 below. We therefore had to compromise on the number and scope of the constructed individuals, finding a balance that allowed us to construct meaningful cases on the one hand, while on the other guaranteeing that we did not create too many or too extensive individuals, which would then be unusable.

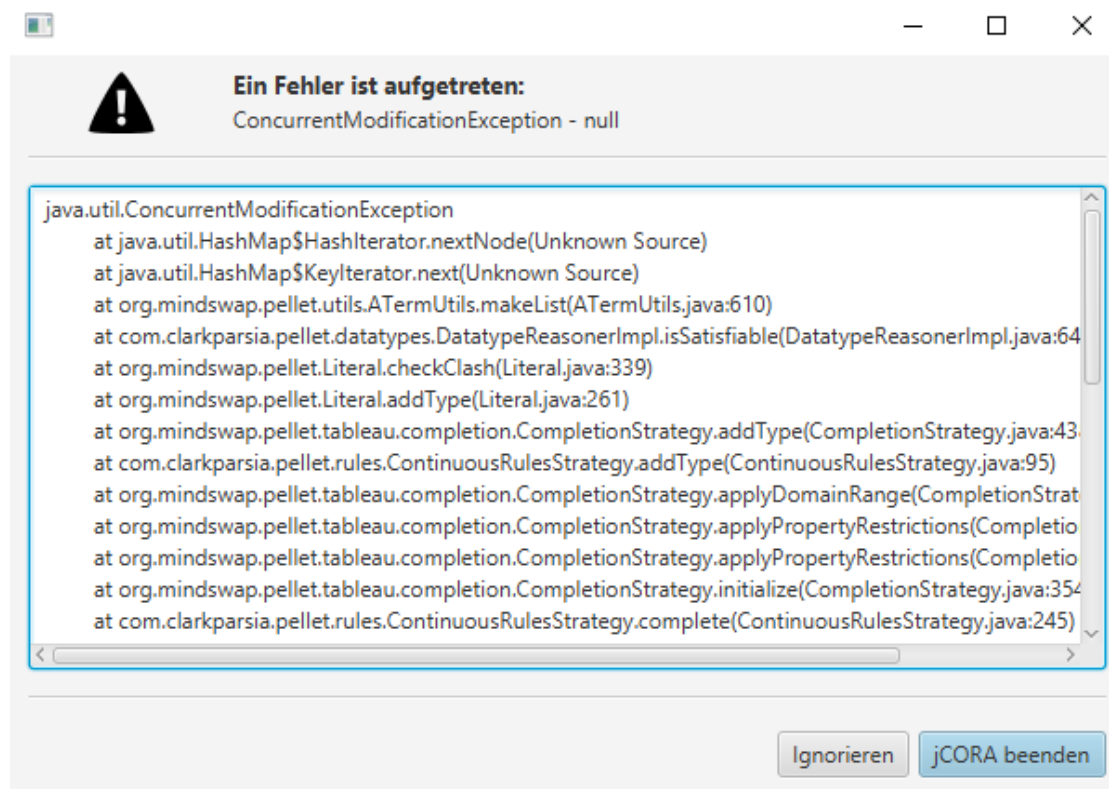


Figure 88: Exemplary error message for the similarity calculation with too many individuals

Strictly speaking, the error message in Figure 88 does not make clear why it occurred. In order to carry out an error analysis, it would be necessary to view the CBR tool jCORa's source code. Yet we assume that this error message is related to the number and scope of the individuals. In addition, jCORa with its case base (project knowledge base) quickly becomes very large when representing knowledge about cases (projects), so that the case base alone takes up several gigabytes of memory space in the case construction carried out in this article. It is also possible to lose cases if string values take

up a larger area when constructing individuals, such as when formulating two or three sentences.

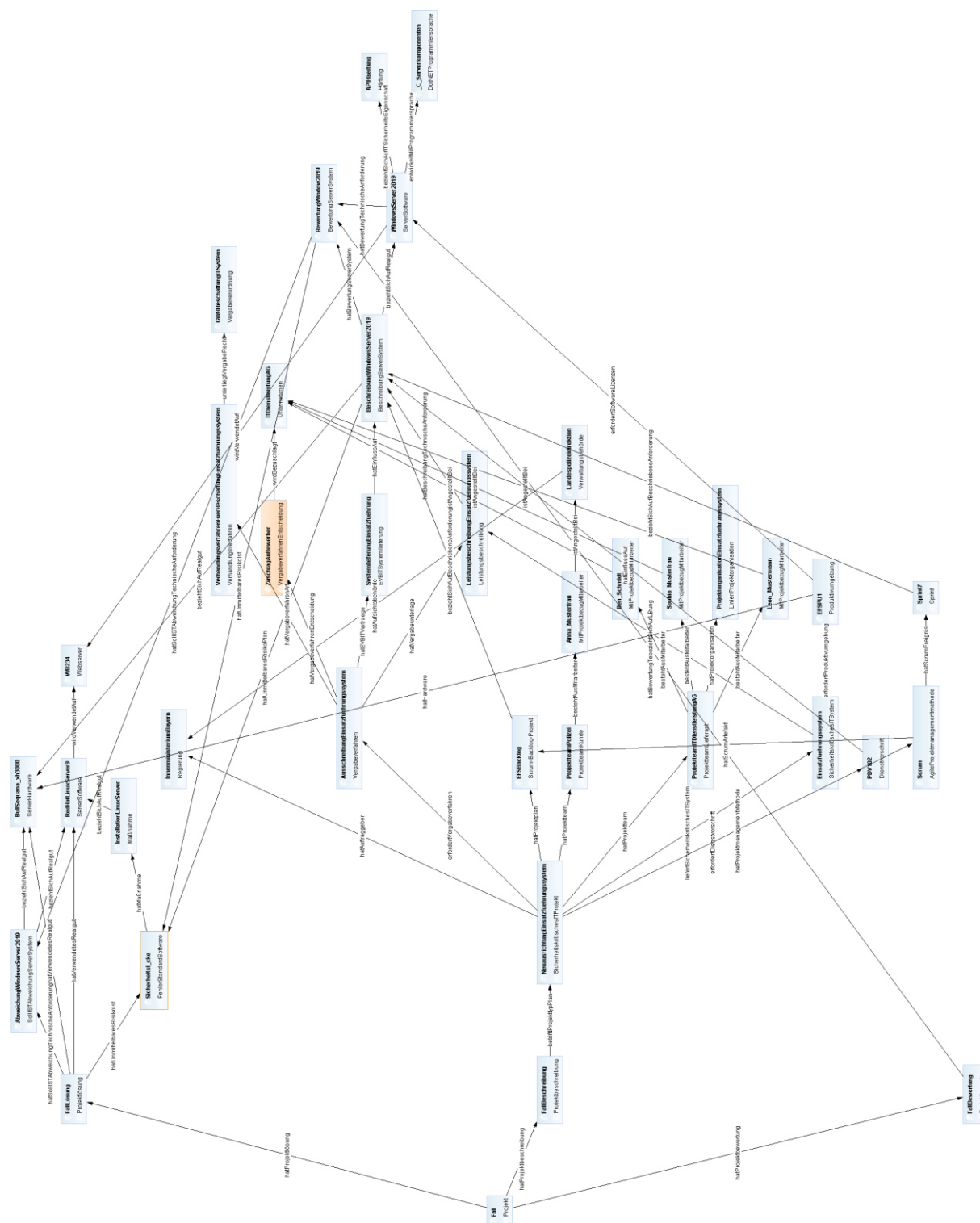
Another problem worth mentioning in case construction is the loss of labeling of the graphical representation using a case graph. This loss of labeling can be corrected by calling the CBR tool jCORA again. However, it is not immediately clear why this error occurs.

In principle, a collection of known errors of the CBR tool jCORA, similar to Protégé, would be desirable in order to have a list of error-causing software problems. These problems would have to be evaluated and rectified in a further problem analysis. Alternatively, the experience gained with the - particularly technical - problems of the CBR tool jCORA could be used to further develop the software on a different technological basis, e.g., as a cloud-native application, in order to avoid running into these problems again.

In this article, we have omitted to make a complete technical evaluation of jCORA: Our primary focus is not on criticizing the CBR tool's problems, but on providing "solution-oriented" further development. We will therefore present the "feasibility" of a reimplementation of the CBR tool jCORA as a cloud-native application later in chapter 4, along with the possibility of providing further specific similarity functions without major adaptations using artificial neural networks for the similarity calculation of words.

3.3.2.2 Case 1: Reorganization of a police command and control system

The case considered here bears the case ID "Neuausrichtung eines Einsatzführungssystems der Polizei" and is an example of a case that extends to an award procedure (expressed by the individual *AusschreibungEinsatzführungssystem*) for the delivery of a command and control system. Figure 89 below illustrates the entire case graph of the case.



der Polizei

The award procedure was carried out as a negotiated procedure. This is expressed by the individual *VerhandlungsverfahrenBeschaffungEinsatzführungssystem*. The attribute *isteuropaweiteausschreibung* with the value *nein* indicates that it was not a Europe-wide

tender. The contract was awarded to the company IT-Dienstleistung AG (individual *ITDienstleistungAG*), which has the attributes *Mitarbeiteranzahl*, *Unternehmenssitz*, *Unternehmensform*, and *Unternehmensname*. As an example, Figure 90 below illustrates the attributes of the individual *ITDienstleistungAG* as well as the non-taxonomic relations this individual has in its range.

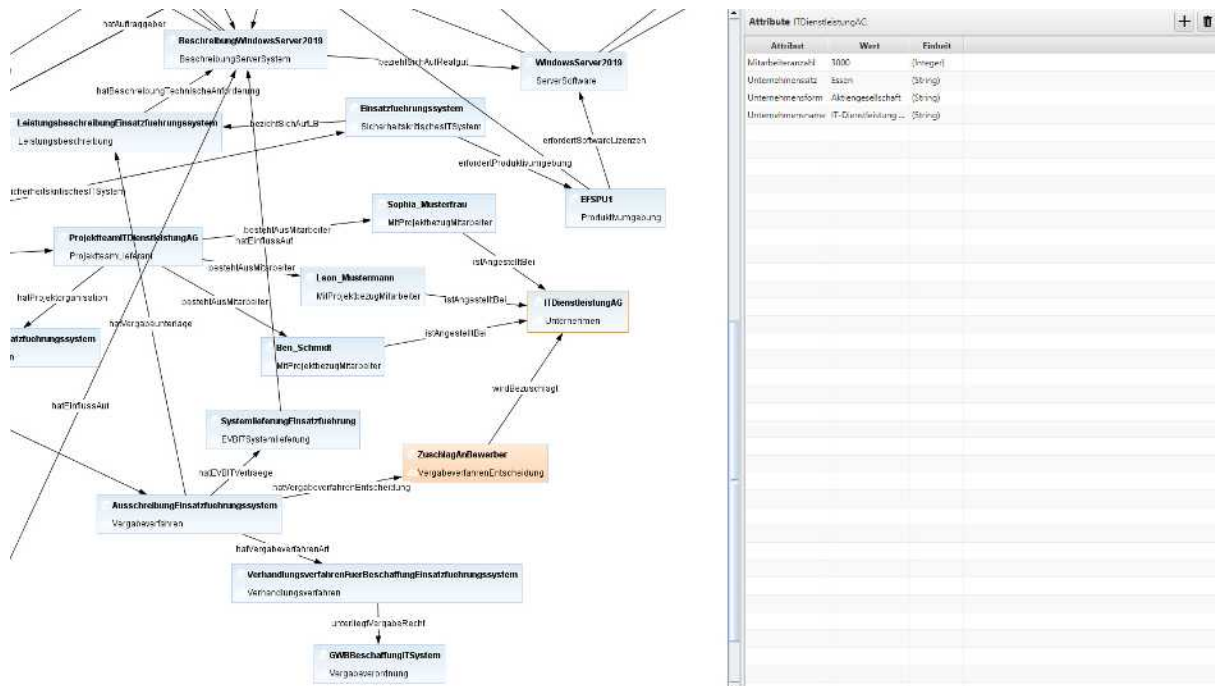


Figure 90: Attributes of the individual *ITDienstleistungAG* and section of the case graph

The requirement that the command and control system must have a server system with Windows Server 2019 software is described in the service description of the award procedure (individual: *LeistungsbeschreibungEinsatzführungssystem*). This is described by the non-taxonomic relation *hatBeschreibungTechnischeAnforderung*. It links the individual *LeistungsbeschreibungEinsatzführungssystem* with the individual *BeschreibungWindowsServer2019*. In order to express the real goods according to the requirements, the non-taxonomic relation *beziehtSichAufRealgut* links the individual *BeschreibungWindowsServer2019* with the individual *WindowsServer2019* and the individual *BeschreibungWindowsServer2019* with the individual *BullSequana_xh30000*. This construction is intended to express that the Windows Server 2019 software is to be provided on the “BullSequana” server hardware for the implementation of this requirement. The described requirement of the Windows Server 2019 to be deployed has a planned (and thus a possible) risk of a security vulnerability, which is expressed by the individual *Sicherheitslücke*. This individual is linked by the individual *BeschreibungWindows*

Server2019 with the non-taxonomic relation *hatUnmittelbaresRisikoPlan*. The measure for remedying the security vulnerability is the installation of a Linux server, which is expressed by the non-taxonomic relation *hatMaßnahme*. The individual *InstallationLinuxServer* is connected to the individual *RedhatLinuxServer9* via the non-taxonomic relation *beziehtSichAufRealgut*, which expresses that a different real good will be used to solve this requirement if the vulnerability in question actually occurs. The non-taxonomic relation *hatUnmittelbaresRisikoIst* shows that this vulnerability risk has actually occurred. It connects the individual *BeschreibungWindowsServer2019* with the individual *Sicherheitslücke*. The individual *SollIstAbweichungWindowsServer2019* deals with the actual deviations in such a way that non-taxonomic relations lead from the individual to the real assets actually used (here *RedHatLinuxServer9* and *BullSequana_xh30000*), and an explanation of the deviation is provided in text form in the attribute *hatErläuterungDerAbweichung*. Figure 91 below illustrates the section of the aforementioned relationships.

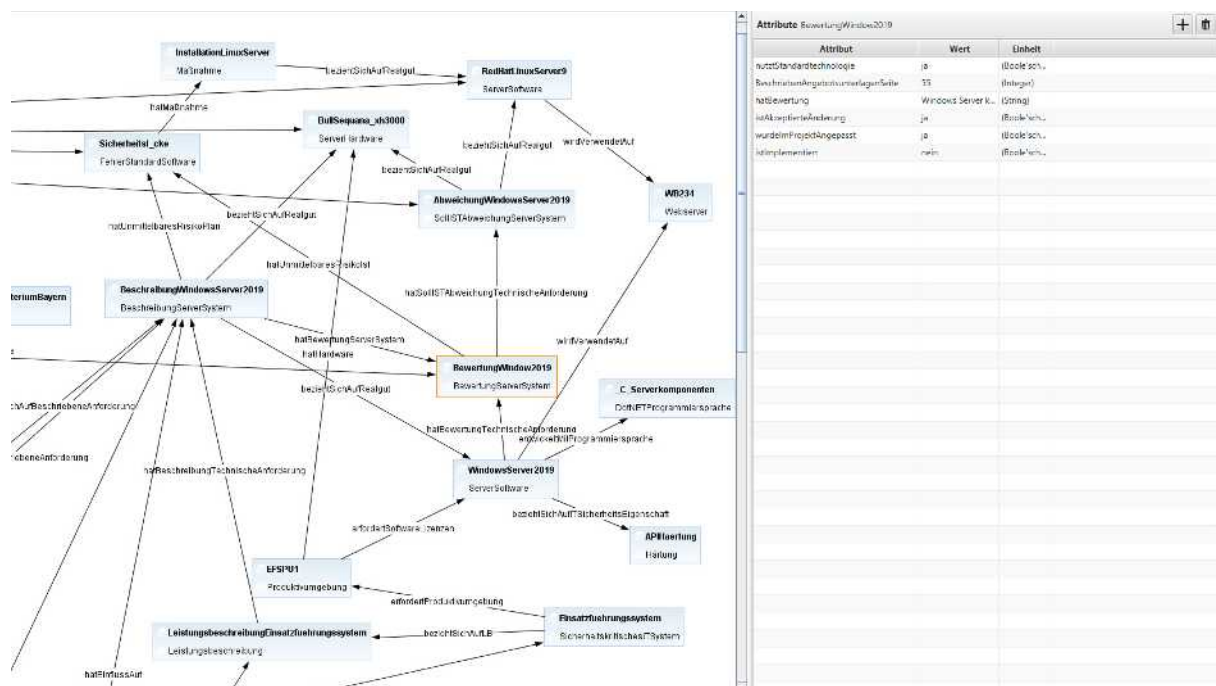


Figure 91: Requirements of a “Windows Server 2019”

The case solution can be constructed in two ways. The first option is to proceed analogously to the *project description*. Here, instead of the non-taxonomic relation *betrifftProjekttypPlan*, which links the individuals *Fallbeschreibung* and *NeuausrichtungEinsatzführungssystem* (from the class *SicherheitskritischesITProjekt*), the non-taxonomic relation *betrifftProjekttypIst* is used. This construction allows access to all linguistic means of expression that are also possible in the *project description*. A second option is

to refer directly to the real assets, risks, and deviation descriptions actually used, as has been done in this case, for example, using local individuals designed only in jCORA.

Figure 92 below shows an example of how the non-taxonomic relation *hatVerwendetesRealgut* is used to refer to the real goods actually used and how the non-taxonomic relation *hatTechnischeAnforderungAbweichung* is used to refer to the deviations in the technical requirements. An analogous procedure is also possible using other relations, e.g., to address the actual project members or project management methods.

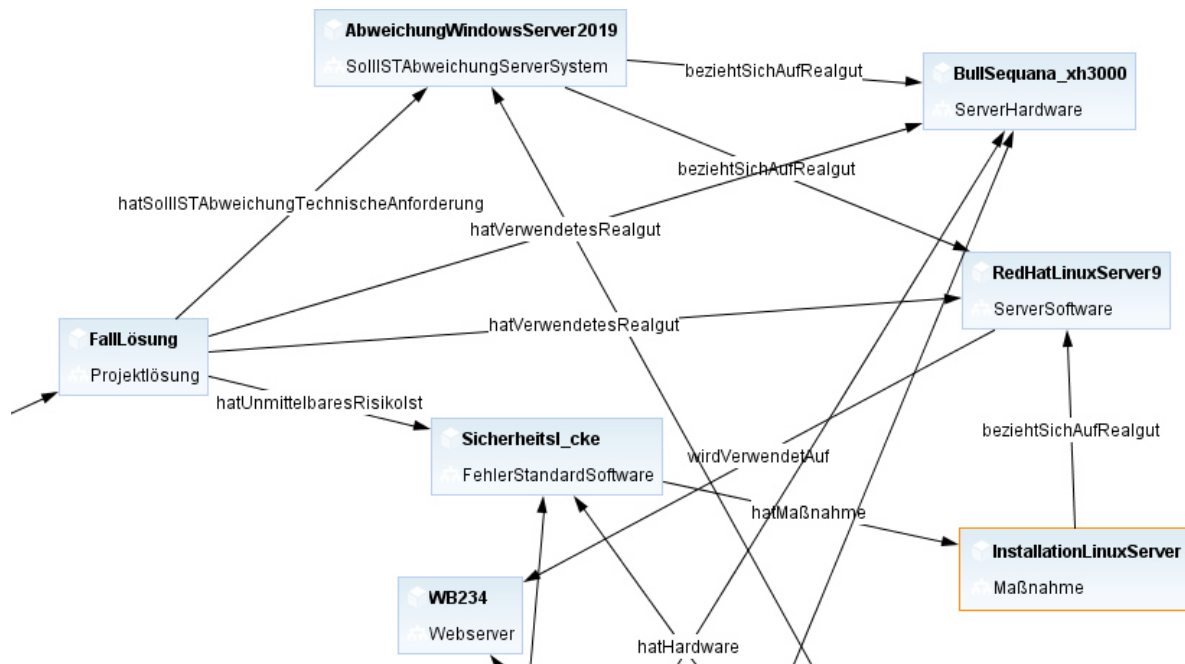


Figure 92: Exemplary case solution for the deviation from the “Windows Server 2019” requirement

3.3.2.3 Case 2: Setting up a cooperative control center

The following case has the case ID “Kooperative Leitstelle”. It is an example of a case in which an award procedure (expressed by the individual *Ausschreibung_Aufbau_Kooperative_Leitstelle*) required the delivery of an operations control system to be used by both the police and the fire department. Figure 93 below illustrates the entire case graph.

by the individual *DellPowerEdge*. This hardware is operated in the productive environment of the fire department (*ProduktivumgebungFeuerwehr*) and accesses freely available geodata called Open Street Maps (*OpenStreetMapDaten*). The command and control system (*EinsatzleitsystemDerPolizeiUndFeuerwehr*) is operated on two productive environments, one for the police and one for the fire department. Both the police and the fire department access the software. However, this is only made available on the hardware in the productive environment of the fire brigade, to which the police also have access. This requirement for situation management harbors the planned risk that user acceptance for situation management could be lacking. This risk did not materialize during implementation, although it was planned to involve a stakeholder, namely the supervisory authority for police and fire departments (*InnenministeriumNRW*), in the event of problems with user acceptance. Deviations were defined in the project, which are expressed by the individual *AbweichungKoorperativesEinsatzmanagement*. For example, it was determined that not all situations have to be handled with joint situation management, but that exceptions can also be defined. An internal note, expressed by the attribute *hatInternenVermerk*, which has been given the data type *String*, describes what has contributed to the compromise—namely that a change management procedure has been applied. The attribute *hatInternenVermerk* is an example of how, although many contexts can be expressed by non-taxonomic relations, in practice many small notes are of great importance because they contain reusable experience-based knowledge. This field, which has a string data type, can be analyzed later—e.g., to calculate similarities of string values using specific similarity functions. The later implemented similarity function for string values can be used for this purpose.

Figure 94 below illustrates the aforementioned situation using a case graph.

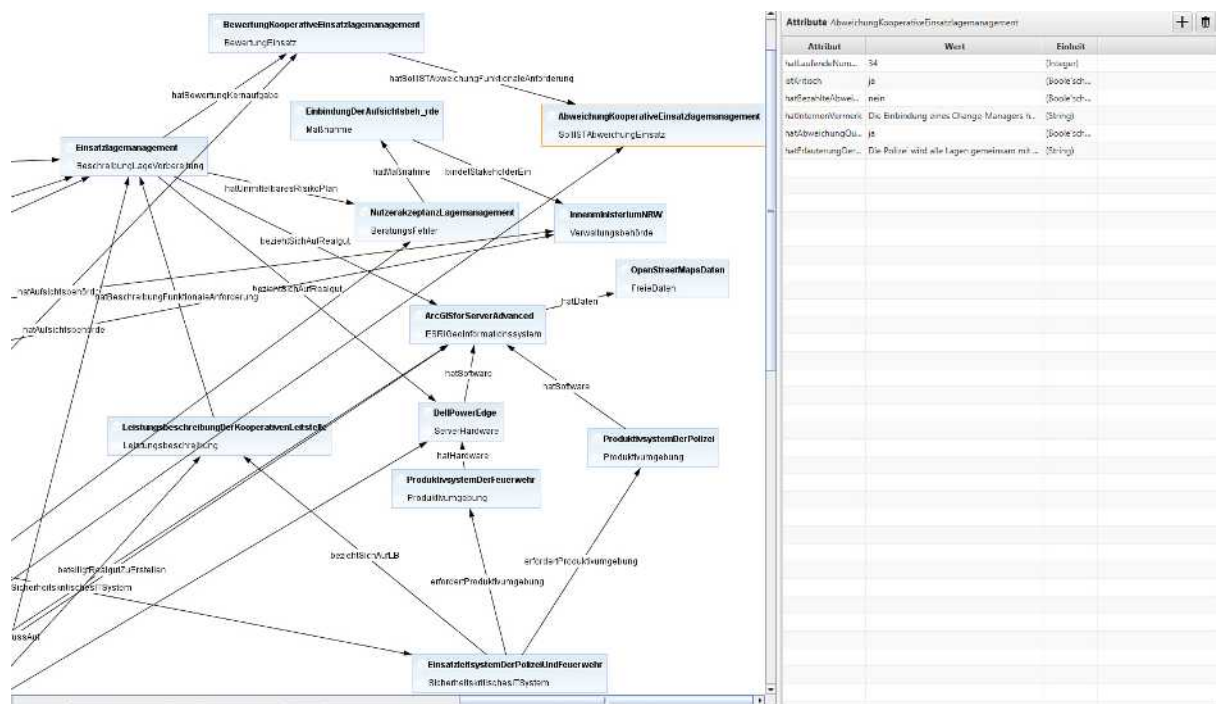


Figure 94: Requirements for joint situation management

The case evaluation has non-taxonomic relations, such as *hatBewertungFunktionaleAnforderung*, which can access the assessment's individuals. This modeling allows direct access to the valuation knowledge of the described requirements. The case solution directly accesses the knowledge components that have contributed to the specific case solution, such as the actual stakeholders, project members, real assets, and deviations between the target and actual values of the described requirements.

The diagram illustrates a complex network of relationships between various entities. The entities are represented as boxes with a title and a subtitle. The relationships are shown as directed arrows with labels. The entities include: BewertungKooperativeEinsatzlagenmanagement, Feuerwehr, AbweichungKooperativeEinsatzlagenmanagement, FallBewertung, FallLösung, Landespolizeidirektion, Karl_MustermannBD, Andrea_BrandORD, Christian_HeinrichPHK, ProjektteamPolizei, ProjektteamFeuerwehr, and ArcGISforServerAdvanced. The relationships include: hatProjektteam, hatVerwendetesRealgut, beziehtSichAufRealgut, hatSoftware, istAngestelltBei, bestehtAusMitarbeiter, hatStakeholderist, hatSchrittAbweichungFunktionaleAnforderung, hatSchrittAbweichungFunktionaleAnforderung, hatProjektmitglied, and hatProjektmitglied.

Figure 95: Exemplary excerpt of the *case solution* and the *case evaluation*

The following case has the case ID “Aufbau einer zentralen Datenbank für Ermittlungen” and is an example of a case in which the delivery of a central database for investigation procedures, which is to be used by the police in all federal states, was requested in an award procedure (expressed by the individual *Ausschreibung Zentrale Datenbank*).

Figure 96 below illustrates the entire case graph.

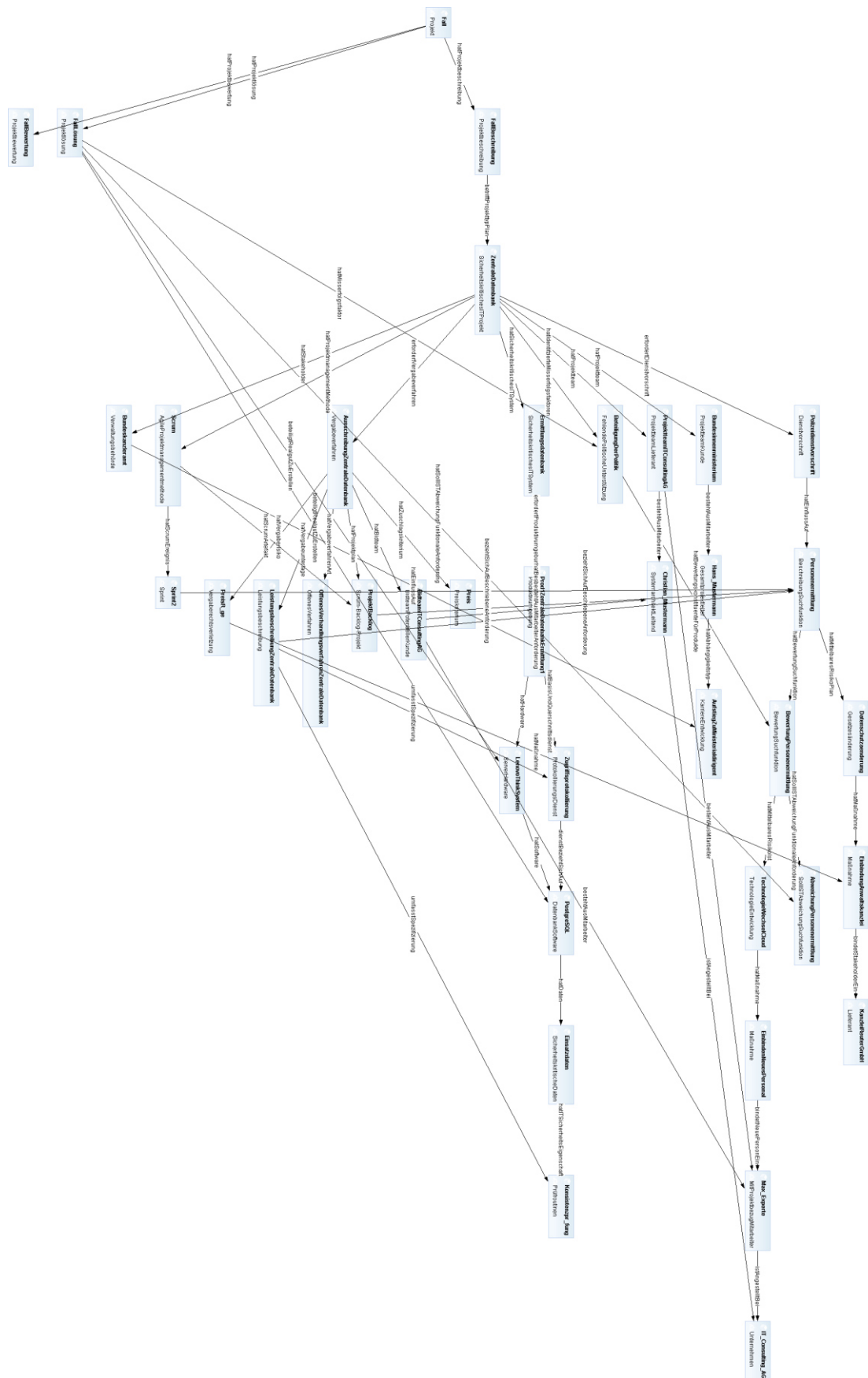


Figure 96: Case *zentrale Datenbank für Ermittlungsverfahren*

The case shows, for example, that there is a procurement risk in the award procedure in the form of a feared breach of procurement law, as price alone decided the procedure's outcome. This could be considered a breach of public procurement law, as it could potentially neglect quality, innovation, and the principle of selecting the best in a cost-effective manner. Inferior services and a distorting effect on competition may result, which a bidder could challenge.

The award criterion is expressed by the individual *Preis*. The non-taxonomic relationship *hatZuschlagskriterium* links the individual *AusschreibungZentraleDatenbank* with the individual *Preis*. To reduce the risk of a breach of public procurement law, a measure has been implemented that is expressed by the individual *EinbindungAnwaltskanzlei*. This measure represents that an external procurement lawyer, expressed by the individual *KanzleiReuterGmbH*, is consulted in order to reduce the procurement risk. Furthermore, the case is that the same person is involved in the award procedure as in the project. Thus, the individual *BidteamITConsultingAG* is linked to the individual *Christian_Mustermann* with the non-taxonomic relation *bestehtAusMitarbeiter* as well as the individual *ProjektteamITConsultingAG*. This connection is interesting insofar as people who are involved in the award procedure and also in the later project have a lower loss of knowledge than people who only join the project later, because the first-mentioned people are involved in the project from the start. It would be conceivable to design a SWRL rule in such a way that the continuous presence of a person from the award procedure can also represent a success factor in the project's later course.

Another fact worth mentioning in this case is the dependency of the customer's overall project manager. The customer's overall project manager is motivated by the fact that successful project completion can lead to a promotion to Ministerial Director. This is expressed by the individual *AufstiegZuMinisterialdirigent*. For this purpose, the individual *Hans_Mustermann*, which represents the overall project manager, is linked to the individual *AufstiegZuMinisterialdirigent* via the non-taxonomic relation *hatAbhängigkeitstyp*. This promotion is influenced by the individual *Bundeskanzleramt*. The individual *ZentraleDatenbank*, which represents the safety-critical IT project, is connected to the individual *Bundeskanzleramt* via a non-taxonomic relation *hatStakeholder*. The value *nein* for the attribute *hatNegativenEinfluss* from the individual *AufstiegZuMinisterialdirigent* expresses that this dependency type has no negative influence on the project. The attribute *hatAbhängigkeitstypErläuterung* contains an explanation of the dependency. The data type is *String*, so that entire sentences can be formulated to explain the dependency.

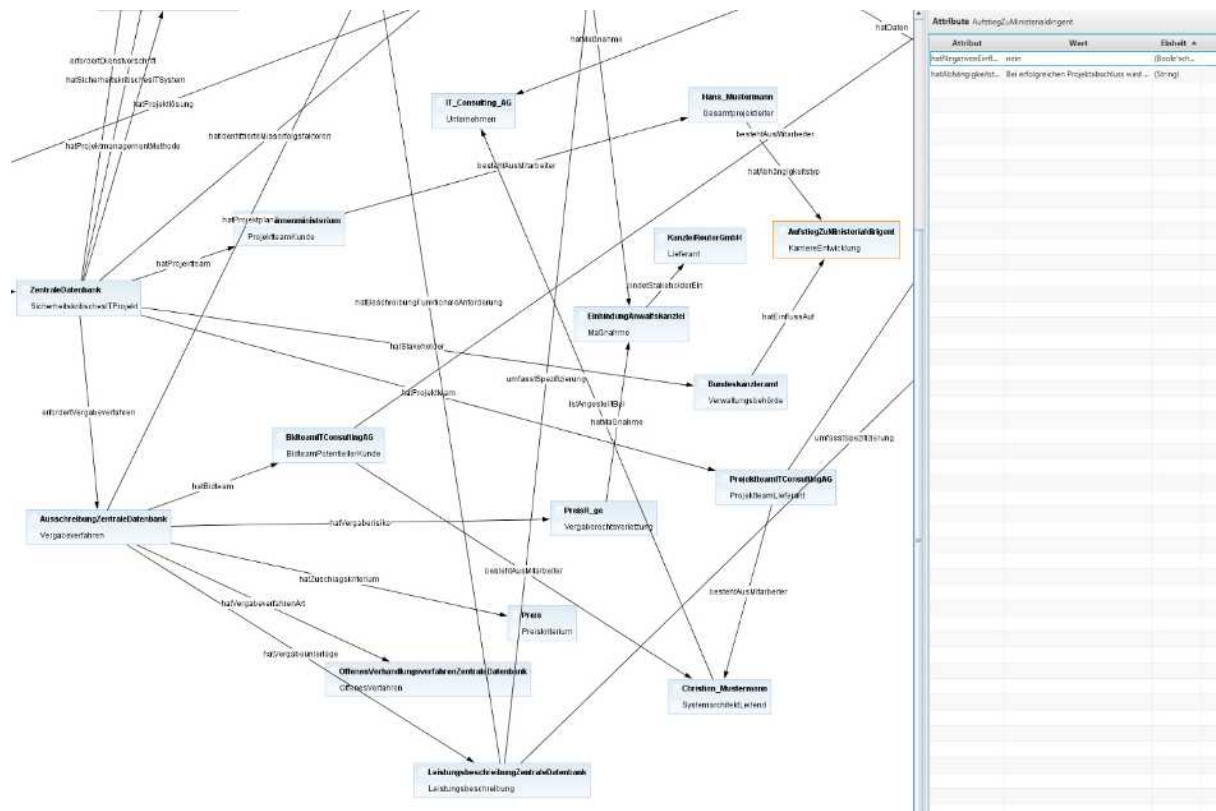


Figure 97: Presentation of the project team

In addition, the case represents a request for a person determination, which is expressed by the individual *Personenermittlung* of the same name. The attribute *hatBeschreibung*, which is of the data type *String*, describes the request from the service description. The attribute *hatAngebotstext*, also of the data type *String*, contains the specific offer text for this request. The individual *Polizeidienstvorschrift*, which is also expressed by the police service regulation of the same name, has an influence on the individual *Personenermittlung*. There also exists an individual *BeteiligungDerPolitik* that influences the *Personenermittlung*, which is linked to the individual *Personenermittlung* via the non-taxonomic relation *hatBeschreibungFunktionaleAnforderung*. This expresses the fact that this project has a failure factor due to a lack of political support, which has a concrete effect on the requirement *Personenermittlung*. At this point, it becomes clear that a clear distinction between risks and failure factors represents a challenge. Failure factors are potential causes for the occurrence of risks that can impair a project's success, or even lead to its failure. One such example is the lack of political support; others include a lack of resources and inadequate planning.

The implementation of the requirement *Personenermittlung* is to be carried out by the real assets *PostgresSQL* and *LenovoThinkSystem*. An individual *Zugriffsprotokollierung*, which documents the logging of accesses, relates to the real asset *PostgresSQL*.

[illegible]

Figure 98: Presentation of a requirement, a risk, and a failure factor

3.3.3.1 Calculation basis of the CBR tool jCORA

The similarity calculations of the CBR tool jCORA are based on the similarity algorithm developed by BEIBEL (2011), pp. 159–173, for ontology-based CBR systems. However, this similarity algorithm suffered from some “technical” limitations; cf. BERGENRODT/KOWALSKI/ZELEWSKI (2015), pp. 479–480. As a result, the algorithm developed by BEIBEL, cannot be used in a general way to calculate similarity in ontology-based CBR

systems. This algorithm was therefore further developed in BERGENRODT/KOWALSKI/ZELEWSKI (2015), pp. 492–511, for the purpose of general applicability and implemented in the CBR tool jCORA. The following explanations refer to this further-developed similarity algorithm.

An ontology's taxonomy is an important starting point for the calculation of similarity in the context of ontology-supported case-based reasoning. The directed paths in an ontology graph, which reflect the taxonomic subsumption relationships between the ontology's classes, as well as the length of these paths make it possible to calculate the similarity using auxiliary functions from graph theory. In addition, the attributes of a class and the non-taxonomic relations between the classes also play an important role in the similarity calculation. We explain these relationships in more detail below using the central constructs of class similarities (or, understood here synonymously, concept similarities), and partial and complete similarities.

Class similarity describes the similarity of two classes k_a, k_b , that belong to the same ontology O . Class similarity is calculated by combining two calculations:

- calculation of the semantic distances between the classes in the graph of the ontology (ontology graph) and
- calculation of the similarities between class properties, which include the attributes of a class on the one hand and the relations in which a class participates on the other.

We will first explain the calculation of semantic distances. This requires the following two auxiliary functions, which are based on graph theory. In both cases, they refer to any two nodes of the ontology graph, each of which represents a class. These two auxiliary functions determine:

- the Least Common Subsumer (LCS) as the “lowest” node, which in the ontology graph is jointly superior to two nodes under consideration, and
- the length of the path between two considered nodes in the ontology graph.

For two classes k_a, k_b that belong to the ontology graph of an ontology, the auxiliary function lcs determines the class k_{lcs} that fulfills both the subsumption property and the minimality property of the Least Common Subsumer class. Put simply, the auxiliary function lcs uses $lcs(k_a, k_b)$ to calculate the class k_{lcs} (LCS class) that firstly represents a superclass of the classes k_a and k_b and secondly has the lowest possible position in the ontology graph of the ontology. The following applies here:

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

The auxiliary function *pfad* determines for two classes k_a and k_b the smallest possible set of classes via which class k_b can be reached in the ontology graph from class k_a on a connected path with edges directed in the same direction. This auxiliary function can be understood as the minimum path length between the classes k_a and k_b in the ontology graph, which enumerates those classes that are traversed on the path with minimum path length. The classes k_a and k_b are counted at the beginning and end of the path in the ontology graph. The following applies:

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

With the help of the auxiliary functions *lcs* and *pfad*, the function *dist* is calculated as the semantic distance between two classes. The *dist* function determines the semantic distance as the maximum length of the two paths of minimum length that extend between the two classes k_a, k_b , which are to be compared in terms of their similarity, and the jointly superordinate LCS class k_{lcs} . The semantic distance can be calculated as a combination of formulas (1) and (2):

$$dist(k_a, k_b) = \max(|pfad(k_a, lcs(k_a, k_b))|, |pfad(k_b, lcs(k_a, k_b))|) \quad (3)$$

The second factor influencing the calculation of class similarity is the similarity between two classes k_a and k_b in terms of their *class properties*—i.e., their *attributes* and their non-taxonomic *relations*. To do this, the class properties of a class are first determined. This is done using the auxiliary function $KE(k_a)$, which is applied here to class k_a as an example:

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

The set $KE(k_a)$ of class properties of a class k_a indicates which properties are either defined as properties of class k_a by explicit specifications or are indirectly “inherited” by this class through the superclasses of class k_a in the sense of object-oriented system design. Using the class properties, the similarity $sim_{ke}(k_a, k_b)$ of two classes k_a and k_b can be calculated with regard to their properties as follows:

$$sim_{ke}(k_a, k_b) = \begin{cases} 0.0 & \text{if } KE(k_a) \cup KE(k_b) = \emptyset \\ \frac{|KE(k_a) \cap KE(k_b)|}{|KE(k_a) \cup KE(k_b)|} & \text{if } KE(k_a) \cup KE(k_b) \neq \emptyset \end{cases} \quad (5)$$

The similarity of two classes with regard to their class properties is therefore calculated by dividing the number of common class properties found in the intersection by the number of all class properties of the two classes to be compared. However, if both sets of class properties are empty, the similarity with regard to the class properties assumes

the value 0.0, because it seems pointless to speak of a (positive) similarity with regard to class properties if there are no class properties at all.

By combining the semantic distance with the calculation of the similarity with regard to the class properties, the class similarity $ksim(k_a, k_b)$ between two classes k_a and k_b can finally be calculated. To do this, the semantic distance between the two classes k_a and k_b is first calculated. The semantic distance's reciprocal value is used to determine the similarity of the two classes on the basis of their semantic distance in the underlying ontology. Subsequently, the similarity of the two classes k_a and k_b is determined with regard to the matching class properties. The following formula (6) shows the determination of the class similarity by combining formula (3) for the semantic distance with formula (5) for the calculation of the similarity with regard to the class properties:

$$ksim(k_a, k_b) \quad (6)$$

$$= \begin{cases} 1.0 & \text{if } k_a = k_b \\ \frac{1}{dist(k_a, k_b)} * \frac{|KE(k_a) \cap KE(k_b)|}{|KE(k_a) \cup KE(k_b)|} & \text{if } k_a \neq k_b \text{ and } KE(k_a) \cup KE(k_b) \neq \emptyset \\ 0.0 & \text{if } KE(k_a) \cup KE(k_b) = \emptyset \end{cases}$$

Individual similarity is a similarity calculation in which the class similarity is first combined with the similarity of the properties defined for the individuals. The result is referred to as *partial* individual similarity. It later serves as the basis for calculating the *complete* individual similarity.

Similarity types, which define how the similarity between two individuals of the same similarity type is determined, are used to calculate the individual similarity. For each similarity type—represented by the parameter n —there is usually a specific similarity function $sim_n(...)$. There may be rare cases where no suitable specific similarity function has been specified for a similarity type. There are several options for this special case. For example, the similarity value 0 (for “completely dissimilar”) can be assumed if no specific similarity function exists for the similarity type. However, we do not consider this situation in the following analysis. It should be emphasized that specific similarity functions play an important role for the similarity calculation and that the availability of such a function specified for a certain similarity type proves to be essential, as otherwise no similarities between individuals with this similarity type can be calculated.

In the following, we assume that for each individual property there is exactly one similarity type that has exactly one specific similarity function. Therefore, we assume a 1:1:1 cardinality for the individual properties, similarity types, and similarity

functions. Specific similarity functions are implemented later in an exemplary manner. We examine the *simStringBOS* function for determining the similarity of words in more detail.

A specific similarity function $sim_n(ie; a; b)$ determines the similarity between the values (attribute or relation values) a and b of an individual property ie that belongs to the similarity type n .

Based on BERGENRODT/KOWALSKI/ZELEWSKI (2015), p. 502, DIVARI (2011), p. 25, STAAB (2011), p. 12, EL JERROUDI (2010), pp. 40–41, and RICHTER (2008), pp. 29–30, the following functional properties apply for each specific similarity function $sim_n(ie; a; b)$:

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

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

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

The calculation of both the partial individual similarity and the complete individual similarity, require several symbols for variables, auxiliary functions, and parameters, as described in the following Table 55.

Symbol	Description
i_a	The variable i_a denotes an individual.
$IE(i_a)$	The auxiliary function IE determines the set $IE(i_a)$ of all properties of the individual i_a .
n	The parameter n represents a similarity type for which a specific similarity function $sim_n(\dots)$ exists.
$IE(i_a, n)$	The auxiliary function IE determines the set $IE(i_a, n)$ of all properties of the individual i_a which have the same similarity type n with the specific similarity function $sim_n(\dots)$.
D	The symbol D defines the set of all similarity types n . Therefore $n \in D$ applies.
ie	The variable ie describes a property of an individual. The individual property can be a relation or an attribute.

w_{ie}	The variable w_{ie} specifies the weighting of the individual property ie .
$wert(i_a, ie)$	The auxiliary function $wert$ determines for the individual property ie the set $wert(i_a, ie)$ of values that are assigned to the individual i_a in relation to this individual property.
IE	The set IE defines the set of all individual properties ie .
$IE_n(i_a, i_b)$	The set IE_n contains all similarity-relevant individual properties of the similarity type n that are assigned to both individuals i_a and i_b together.
$K(i_a)$	The auxiliary function K determines the class $K(i_a)$ of the individual i_a .

Table 55: Variables, auxiliary functions, and parameters
for calculating individual similarities

Based on the variables, auxiliary functions, and parameters explained in Table 55, the *partial individual similarity* can first be determined. For this purpose, all their common individual properties of similarity type n are determined for the individuals i_a and i_b to be compared:

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

If $IE_n(i_a, i_b) = \emptyset$ applies to this set of common individual properties, the individual similarity with regard to the similarity type n is equal to 0.

The similarity between two individuals i_a and i_b is calculated with respect to all similarity types n and all common individual properties using the following property-based *partial* similarity function $esim(i_a, i_b)$:

$$esim(i_a, i_b) \quad (8)$$

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

with:

$$h(n, ie, a, b)$$

$$= \frac{\sum_{a \in wert(i_a, ie)} \left(\max_{b \in wert(i_b, ie)} sim_n(ie, a, b) \right) + \sum_{b \in wert(i_b, ie)} \left(\max_{a \in wert(i_a, ie)} sim_n(ie, b, a) \right)}{|wert(i_a, ie)| + |wert(i_b, ie)|}$$

According to formula 8, for each similarity type n and for each associated individual property ie from the set IE_n , the maximum similarity $sim_n(ie, a, b)$ with respect to each element a from the set $wert(i_a, ie)$ of values of the individual property ie defined for the individual i_a is determined on the one hand for each element b from the set $wert(i_b, ie)$ of values of the individual property ie defined for the individual i_b using the specific similarity function sim_n of the type n .

On the other hand, for each element b from the set $wert(i_b, ie)$ of values of the individual property ie defined for the individual i_b , the maximum similarity $sim_n(ie, b, a)$ with respect to each element a from the set $wert(i_a, ie)$ of values of the individual property ie defined for the individual i_a is also determined using the specific similarity function sim_n of the type n .

Put simply, each value a of the individual property ie for the individual i_a is compared sequentially with the values b of the individual property ie for the individual i_b . The maximum similarity value is selected according to the values $sim_n(ie, a, b)$ of the specific similarity functions sim_n for the similarity types n .

The procedure then proceeds in the same way by swapping the individuals i_a and i_b . The sums determined for all values a and b of the individual property ie are divided by the numbers of all individual property values a and b from the sets $wert(i_a, ie)$ and $wert(i_b, ie)$ respectively; cf. BERGENRODT/KOWALSKI/ZELEWSKI (2015), p. 505. By normalizing the previously calculated sums, an average similarity is determined for each individual property ie and then multiplied by the weight w_{ie} that the individual property ie has from the perspective of the users of the CBR tool jCORA.

Using the partial individual similarity explained above and the class similarity presented earlier, the individual similarity can be calculated for two individuals i_a and i_b completely. The *complete individual similarity* for two individuals i_a and i_b can be determined using the function *isim* by multiplicatively linking the two similarity calculations already presented with regard to class similarity and partial individual similarity. In addition, the partial individual similarity is normalized by the sum of the weights of all properties ie of the individuals i_a and i_b . Therefore, for the function *isim* of complete

individual similarity for the normal case, the sets of individual properties for the two compared individuals i_a and i_b are not empty and both individuals i_a and i_b have at least one common individual property:

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

if $IE(i_a) \neq \emptyset$ and $IE(i_b) \neq \emptyset$ and $IE(i_a) \cap IE(i_b) \neq \emptyset$

Two special cases must be taken into account when calculating individual similarity.

The first special case occurs if no individual properties are defined for the two individuals i_a and i_b . The union of both individual property sets $IE(i_a)$ and $IE(i_b)$ would then be empty. Here, the normalized partial similarity between the individuals i_a and i_b —i.e., the second factor in the product of formula 9—is set to the value 1:

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

The second special case occurs if the union of the two individual sets $IE(i_a)$ and $IE(i_b)$ is not empty, but no individual properties are defined for one of the two individuals i_a or i_b . Here, the normalized partial similarity between the individuals i_a and i_b —i.e., the second factor in the product of formula 9—is set to the value 0:

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

3.3.3.2 Exemplary similarity calculation using the CBR tool jCORA

As an example, we perform a “manual” similarity calculation for the two projects *RegionalleitstellenverbundSchleswigHolstein* and *KooperativeLeitstelleBerlin*. As individuals, these two projects belong to the classes *SicherheitskritischesITProjekt* and *Vergabeverfahren* from the safety-critical IT project ontology. The similarity calculation is therefore—as is usual with ontology-supported CBR systems—primarily located at the ontology’s individual level, including the individual properties, but also uses its classes with their class properties.

We consider the following simplified excerpt from the safety-critical IT project ontology in order to be able to carry out the similarity calculation clearly.

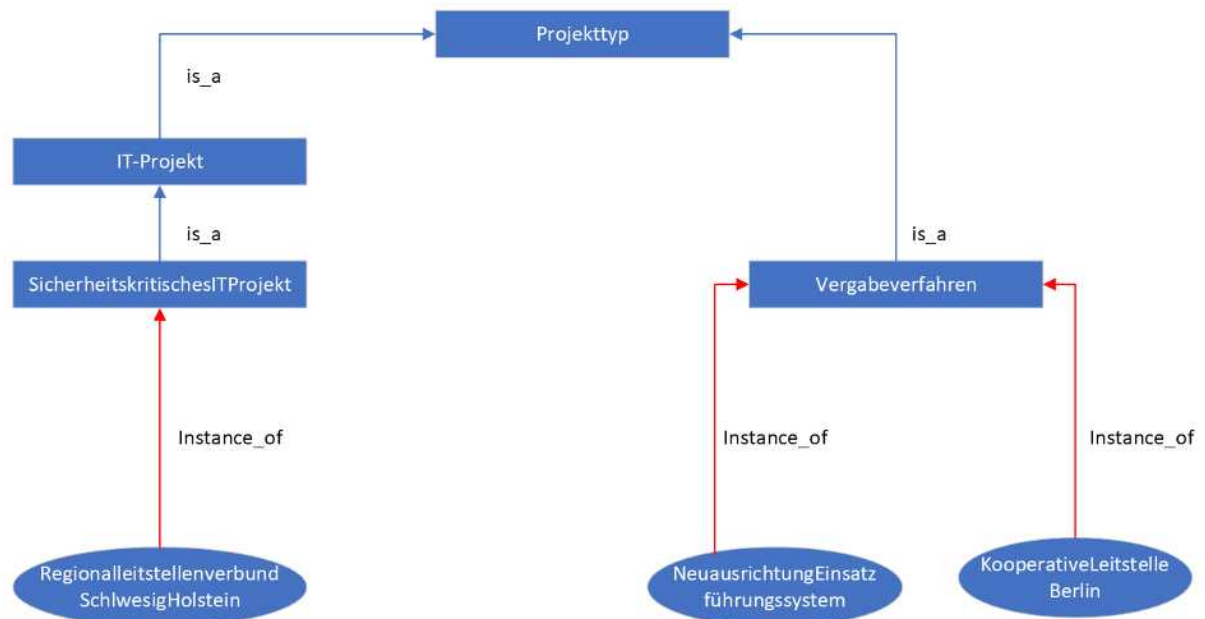


Figure 99: Excerpt from the safety-critical IT project ontology

The four individual properties of each of the three individuals in Figure 99 above can be seen in Figure 100 below. The specific values of the individual properties are also displayed.

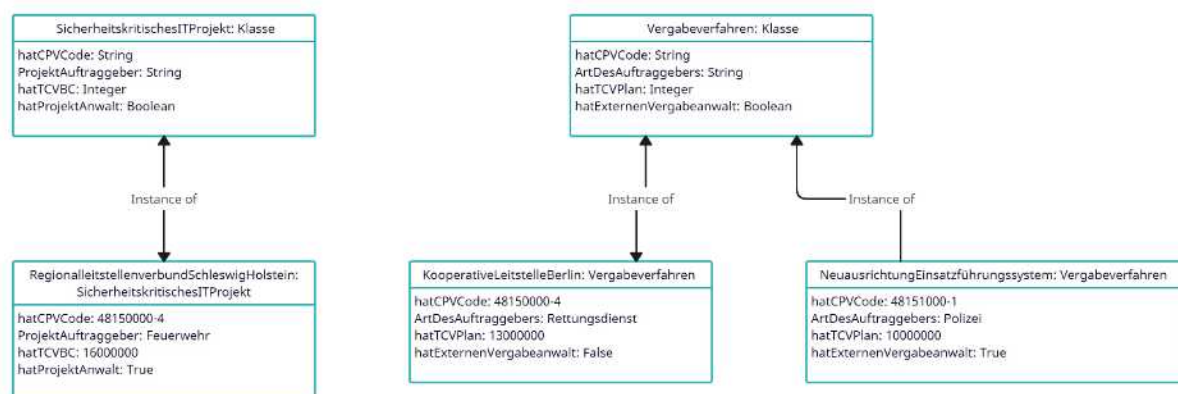


Figure 100: Representation of the values of the individual properties

The calculations for the two individuals (projects) *RegionalleitstellenverbundSchleswigHolstein* and *KooperativeLeitstelleBerlin* proceed in the following order: class similarity (*ksim*), partial individual similarity (*esim*), and finally complete individual similarity (*isim*).

The results of the similarity calculations using the CBR tool jCORA based on the three cases presented in sections 3.3.2.2 to 3.3.2.4 are presented below. They concern Case 1, *Neuausrichtung_eines_Einsatzführungssystems_der_Polizei*, Case 2, *Aufbau_Kooperative_Leitstelle*, and Case 3, *Aufbau_einer_zentralen_Datenbank_für_Ermittlungen*. As we have already explained the similarity algorithm in detail in chapter 3.3.3.1, we do not do so again below.

The similarity calculations are based on Case 2, *Aufbau_Kooperative_Leitstelle*. They take 1 minute and 13 seconds. If three cases that do not include the entire case knowledge of the respective project concerned already require calculations over a minute, it can be assumed that the runtime will increase significantly with complete case knowledge and significantly more than three cases in the case base. In operational practice, this could lead to a loss of acceptance of the CBR tool jCORA.

The result in relation to Case 2, *Aufbau_Kooperative_Leitstelle*, is as follows:

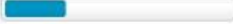
Fall-ID	Ähnlichkeit		Adaptieren	Anzeigen
Aufbau_Kooperative_Leitstelle	100%		Adaptieren	Anzeigen
Neuausrichtung_eines_Einsatzführungssy...	36%		Adaptieren	Anzeigen
Aufbau_einer_zentralen_Datenbank_f_r_E...	29%		Adaptieren	Anzeigen

Figure 101: Similarity calculation for Cases 1 to 3 using jCORA

As jCORA lacks specific similarity functions, the calculated similarity values must be assessed critically. However, the calculated similarity values provide an initial estimate of the similarity between the three safety-critical IT projects.

It is true that the security-critical IT projects *Aufbau_Kooperative_Leitstelle* (Case 2) and *Neuausrichtung_eines_Einsatzführungssystems_der_Polizei* (Case 1) are classified as more similar than the security-critical IT projects *Aufbau_Kooperative_Leitstelle* (Case 2) and *Aufbau_einer_zentralen_Datenbank_für_Ermittlungen* (Case 3).

The similarity between Case 2 and Case 1 is given as 36%. The similarity value is justified by the fact that Case 2 and Case 1 both provide for incident command and control systems as a delivery item, and that they have similar requirements and award procedures. However, Case 2 involves pursuing a cooperative approach between the police and fire department, whereas in Case 1 the system is used exclusively by the police. The assessment that the two cases are not very similar therefore seems plausible.

The similarity between Case 2 and Case 3 is given as 29%, which is also plausible, as both the project management method and the basic structure of the award procedure show similarities. A certain similarity can also be assumed here, but not to the same extent as between Case 2 and Case 1, for which the similarity is calculated at 36%.

Even if the similarity values can be regarded as tendentiously correct, the concrete similarity values of 36% and 29% appear to be too low. This could stem from various reasons, such as the incomplete case presentation: The cases presented may not contain all relevant case knowledge, which could lead to reduced similarity scores as some similarities were not captured. In addition, we weighted all individual properties equally. We did so because we discuss the similarity calculation in the following sections, and equal weighting provides a robust basis to enable comparisons between cases, ensuring the comparability of cases regardless of their specific individual properties. While this facilitates the interpretation and communication of results, it does not adequately reflect reality. We have deliberately accepted this discrepancy, as our main focus remains the fundamental feasibility of a similarity calculation using the CBR tool jCORA.

4 Design of an ontology-supported case-based reasoning system as a cloud-native application

4.1 Preliminary remarks on the system design

We present a conceptual approach to how an ontology-based CBR system can be implemented in a cloud environment. For this purpose, we implement selected functions as serverless functions, using the development environments Cloud9 from Amazon Web Service and Google Colab from Google for this purpose. AWS Cloud9 and Google Colab are cloud-based integrated development environments that make it possible to write, execute, and debug the code exclusively in the browser. For a more detailed explanation of the Cloud9 development environment, please refer to AMAZON WEB SERVICES, INC. (2022b); for an explanation of the Google Colab development environment, please refer to GOOGLE (2022).

In principle, various approaches are available for providing software (“application”) such as the CBR tool jCORA in a cloud environment. The most common approaches for transferring existing monolithic applications consist of transferring an application to the cloud (cloud-enabling) or developing and deploying it from scratch in a cloud (cloud-native); cf. GONIWADA (2022), pp. 17–26; HENNEBERGER (2016), pp. 12–13.

If the existing application is only transferred to the cloud, the technological limitations of the monolithic software structure can often not be resolved or only partially resolved. Furthermore, technological developments can only be used to a limited extent, as software libraries that are available in the cloud can only be used in a monolithic application with a great deal of adaptation. Moreover, the advantages of scalability can also only be realized for the entire application and not for individual functions if it is transferred exclusively to the cloud. For a more detailed explanation of the advantages of cloud-native applications compared to cloud-enabling, please refer to GONIWADA (2022), pp. 19–20. Pure cloud-enabling of an application does not create any added value in terms of innovation, deployment speed, and user experience; cf. LÜNENDONK (2021), p. 14.

The following explanations focus exclusively on the procedure for a new implementation as a cloud-native application. In the long term, this is the most viable option for transferring an application to the cloud in order to be able to utilize the technological advances that result from the cloud now, and could in the future; cf. GONIWADA (2022), p. 17; HENNEBERGER (2016), p. 13.

The aim is to overcome the monolithic character of the CBR tool jCORA. Its functions are implemented as serverless functions in a cloud environment, meaning they operate exclusively in said environment. They can be called up using a web-based user interface.

The following added benefits are to be achieved by designing a cloud-native application in addition to breaking up the monolithic application structure:

- Scalability of the application for the operational application purpose
- Improving the maintainability of the application through the clear separation of the serverless functions and the user interface
- Increasing user acceptance through a user-friendly user interface
- Use of freely available AI development libraries
- Combining the strengths of different cloud environments

In the following, we describe conceptually how ontology-supported case-based reasoning can be implemented as a cloud-native application. However, we have not created a full prototype. Instead, the prototypically developed functions and the user interface designed as a click prototype serve to demonstrate the functionality of ontology-supported case-based reasoning as a cloud-native application in an exemplary manner (proof of concept). Furthermore, the advantages of a cloud-native application are to be demonstrated by using other AI development libraries. For this purpose, we implement a specific similarity function as an example, one that uses artificial neural networks to calculate the similarities between string values for individual properties.

4.2 Cloud environments

Currently, the cloud market for freely accessible cloud environments is essentially dominated by three providers; cf. SYNERGY RESEARCH GROUP (2022). They are also referred to as “hyperscalers”. Amazon Web Service (AWS) is the market leader with a 34% market share. Microsoft Azure follows with a 21% market share. Google Cloud Platform is in third place with a 10% market share. Together, the three providers account for 65% of the global cloud market. Although other smaller providers exist, such as Alibaba Cloud and IBM Cloud, we do not discuss them in detail below.

Cloud environments are significantly transforming current information technology. According to a study by FORTUNE BUSINESS INSIGHTS (2020), the predicted global cloud technology market volume amounted to USD 677.95 billion in 2022. The market is expected to grow to USD 2432.87 billion by 2030. Although the coronavirus pandemic accelerated this growth, it is mainly due to increasing digitalization. An important factor for potential growth is seen in the integration of artificial intelligence (AI) technologies in the cloud.

Due to the cloud's increasing use, the volume of data it holds is also growing rapidly, and must be processed using analysis tools; cf. PERTLWIESER (2022), p. 33. AI technologies support such data analysis. This creates a strong interaction between the cloud and AI technologies. Hyperscalers have various publicly usable building blocks of AI techniques, for example in the form of development libraries. Accordingly, researchers are paying growing attention to the use of AI techniques in cloud-based developer services, which are being examined separately in various studies; cf. PERTLWIESER (2022), p. 33. One study that specifically examines the development possibilities of AI techniques in the cloud, for example, is the special report GARTNER (2022a) on Cloud AI Developer Services.

Two perspectives are therefore important when considering hyperscalers: In addition to the use of a cloud environment, the cloud environments should be accessible for developments, particularly in the field of AI technologies.

Amazon Web Service (AWS) was the first cloud provider on the market (existing since 2006) and is currently the market leader in the cloud sector; cf. BÖGELSACK et al. (2022), pp. 8–9. Amazon Web Service offers its cloud on globally distributed data centers (currently 34 locations), which are represented on all continents except Antarctica; cf. AMAZON WEB SERVICES, INC. (2022e). Amazon Web Services is a subsidiary of Amazon and the company's most profitable business division. The cloud environment now generates more than half of the company's operating profit; cf. AMAZON (2022), pp. 64–65.

In addition to its own services, Amazon Web Service (AWS) offers the option of offering cloud-native applications developed by customers via the cloud environment on special marketplaces and thus making them available to other cloud users. Some of the best-known customers who offer their services on the basis of Amazon servers include Netflix, Disney+, Delivery Hero, LinkedIn, Facebook, and Twitter; cf. AMAZON WEB SERVICES, INC. (2022c).

The Google Cloud Platform (GCP) is part of the Google Cloud, which Google uses to provide its own services, such as YouTube and Google Maps; cf. BÖGELSACK et al. (2022), p. 13. The Google Cloud Platform has existed since 2008. Similarly to Amazon, Google offers access to various Google software products in its cloud environment; cf. GOOGLE CLOUD (2022a). The Google Colab development environment is particularly important for the development of a cloud-native application using a common programming language. Google offers its cloud environment in globally distributed data centers (currently 34 locations), which—with the exception of Africa and Antarctica—are represented on all continents; cf. GOOGLE CLOUD (2022b). Although Google only entered the cloud market after Amazon (AWS) and Microsoft (Azure), it has the strongest

growth amongst all cloud providers, at a rate of 46% in the first quarter of 2021; cf. REGENFUß/NINK (2022).

The Google Cloud Platform stands out as the pioneer in the areas of big data and artificial intelligence. This is based on Google's history in the field of search engines; cf. BÖGELSACK et al. (2022), p. 13; REGENFUß/NINK (2022). In various use cases, the Google Cloud Platform is more frequently used in multi-cloud scenarios as a secondary provider or as a supplement for specialized solutions; cf. REGENFUß/NINK (2022). In addition, unlike other cloud providers, Google has its own high-availability network (including submarine cables) with speeds of up to 250 TBit/s, connecting the continents; cf. GOOGLE CLOUD (2020). Google's own network plays a key role in ensuring low latency and high redundancy

Microsoft launched its cloud under the current name "Microsoft Azure" back in 2008. Similarly to the two cloud providers mentioned above, Microsoft makes several Microsoft products available in the cloud; see MICROSOFT (2023b). As a large number of companies use Microsoft products, the integration of Microsoft products with cloud services is very obvious. Several studies and sources also view this as a strength of Microsoft's cloud; cf. BÖGELSACK et al. (2022), p. 11; GARTNER (2022b). Users of Microsoft Azure can thus continue to use their existing Microsoft licenses in the Azure Cloud or receive a discount if special licensing is necessary. Microsoft provides the Microsoft Azure Cloud in various data centers. The data centers are located on all continents except Antarctica; cf. MICROSOFT (2023a).

In principle, the three aforementioned cloud providers offer high availability of their cloud environments as well as numerous products for the use of cloud services. To design an ontology-based CBR system as a cloud-native application, we selected two cloud providers in whose cloud environments exemplary serverless functions are being developed for use as a cloud-native application. Such a multi-cloud environment offers advantages over a single cloud provider. These include in particular:

- Using the different strengths of the various cloud providers
- Less dependence on a single cloud provider
- Higher availability and reliability through redundancies

This is an example of how the specific strengths of individual cloud providers can be utilized in a multi-cloud environment. A similar approach can also be found in KUNSCHE/SPITZ/POHLE (2022), pp. 403–408.

One cloud provider is selected as the primary provider for the design of ontology-supported case-based reasoning as a cloud-native application. The second cloud provider

serves as a secondary provider. The primary provider implements the majority of the serverless functions. The secondary provider implements special serverless functions in order to utilize individual advantages of its cloud platform.

The following criteria are used to select the two cloud providers:

- Availability of the cloud environment
- Costs of use
- Intuitive usability of the cloud
- Market share in the cloud market
- Future-proofing of the cloud
- Browser-based development environment
- Openness to innovation, especially for the support of AI technologies
- Possibility of providing data exclusively in German or European data centers

We selected the Amazon Web Service (AWS) and Google Cloud Platform (GCP) platforms. We used Amazon Web Service as the primary platform, in which the essential functions are implemented (e.g., reading in an ontology, accessing the ontology, the similarity algorithm, and specific similarity functions). The Google Cloud Platform only implements specific similarity functions that are developed on the basis of artificial neural networks. We used the Google Cloud Platform for the design of the ontology-based CBR system as a cloud-native application as a secondary provider for specialized solutions in the field of artificial neural networks.

Figure 102 below illustrates the use of the two cloud providers as primary and secondary providers.

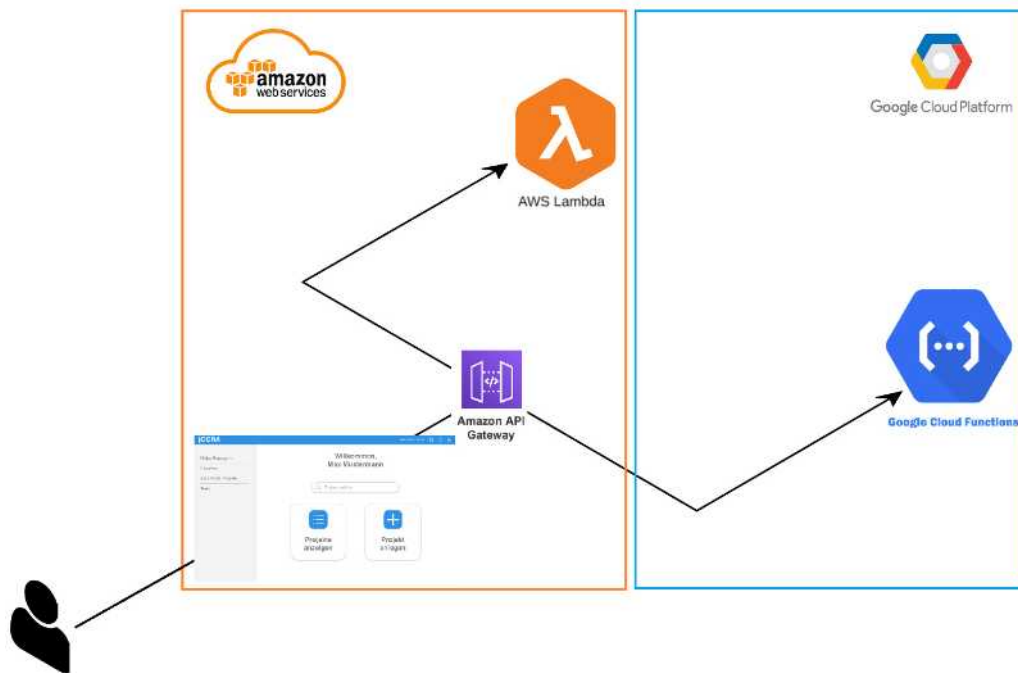


Figure 102: Designed multi-cloud environment

We will first explain our selection of Amazon Web Service (AWS) as the primary provider:

- Amazon Web Service is the market leader with a market share of 34%. This means that Amazon Web Service provides over a third of the cloud environments used worldwide, with a gap of around 10 percentage points to Microsoft Azure; cf. SYNERGY RESEARCH GROUP (2022).
- Amazon Web Service guarantees an availability of 99.9%; cf. AMAZON WEB SERVICES, INC. (2022b).
- Amazon Web Service enables the selection of different locations for the provision of the implemented serverless functions. When deployed in a German or European AWS location, the developed functions are subject to German data protection regulations.
- Numerous publications have criticized the pricing of the various cloud providers; cf. GARTNER (2022b); LINTHICUM (2022); URBAN/GARLOFF (2022), p. 617; GLEB (2021), p. 53. However, Amazon Web Service offers free use for low loads, which is suitable for a conceptual investigation (as in this “proof of concept”).

This does not incur any costs for the prototypical implementation of the individual serverless functions in the AWS cloud.

- The Cloud9 development environment from Amazon Web Service is a fully integrated, browser-based development environment in the AWS cloud that enables applications to be developed using common programming languages and implemented as serverless functions.
- Specialist literature often praises the AWS cloud's use as intuitive; cf. BÖGEL-SACK et al. (2022), p. 123; POTHECARY (2021), p. 111. In addition, there exist much documentation and a wide range of introductory literature on Amazon Web Service's use; cf. for example AMAZON WEB SERVICES, INC. (2022d).

Overall, Amazon Web Service, as the market-leading cloud provider, is best suited as the primary cloud provider for the development of a cloud-native application, particularly due to its pricing for prototype implementations and its fully integrated development environment.

We selected the Google Cloud Platform as a secondary cloud provider in order to utilize the aforementioned advantages of a multi-cloud environment. In particular, the use of development libraries for AI techniques and the extensive documentation using Google Cloud Platform's browser-based development environment represent a key strength of this cloud provider. GARTNER (2022a) also regards the Google Cloud Platform as the leading cloud environment in the field of AI technologies.

The freely available code examples on the Google Cloud Platform make it easy for developers to use large amounts of data in real time from different sources and to understand how to use the AI development libraries. The AI development libraries were created by Google itself, are intended for use as development libraries in the "modern" Python programming language (we will come back to this later), and are freely available for public use. The AI development library "Word2Vec" is particularly important for this study, as we will discuss in detail later. Overall, the special advantage of the Google Cloud Platform is seen in the well-documented AI techniques illustrated with code examples.

In summary, we felt that Amazon Web Service, as an established provider and due to its aforementioned advantages, is suitable as the primary cloud provider for the design of an ontology-based CBR system as a cloud-native application. For the implementation of individual serverless functions based on TensorFlow and Word2Vec (explained later), the Google Cloud Platform is suitable as a secondary cloud provider. This justifies the design of an ontology-supported CBR system as a cloud-native application using a multi-cloud environment.

4.3 Designing an ontology-supported case-based reasoning system as a cloud-native application

4.3.1 Designing the frontend

A concept for the frontend—the user interface (UI)—for an ontology-supported CBR system as a cloud-native application already exists. It was presented in WEBER et al. (2023), pp. 37–102, as a “click prototype”, with the NIELSEN usability engineering process used as the basis for prototype development. The most important results from the aforementioned publication are briefly explained below.

In the first step, the heuristics based on NIELSEN. NIELSEN recommends ten heuristics for carrying out a heuristic evaluation in order to identify usability problems in an application. These heuristics cover problem categories that should be considered when designing an application. The authors supplemented NIELSEN’s ten heuristics with two additional ones. These twelve heuristics in total form the basis for the usability evaluation of the prototypical CBR tool jCORA discussed so far, which serves as the starting point for the design of the front-end and back-end of an ontology-supported CBR system as a cloud-native application. The usability evaluation is carried out by experts who put themselves in the role of an (end) user. The experts examine a system for violations of the heuristics. The twelve heuristics reflect the desired characteristics of the interaction between an (end) user and a system. If a violation of one of these desired characteristics is identified, this is an indication of a possible usability problem. The aim of heuristic evaluation is to identify all usability problems. Priority is given to those usability problems that have a restrictive influence on a system’s usability. This system is represented here by the CBR tool jCORA. The results of the heuristic evaluation are presented in Table 56 below:

	Heuristics	Usability problem (yes/no)	Problem naming
Heuristics from NIELSEN	1. Visibility of the system status	no	
	2. Correspondence between the system and the real world	yes	The terms “class”, “relation”, “non-taxonomic relation”, “individual”, and “attribute” are not ones familiar to a project manager.

	3. User control and freedom	no	
	4. Consistency and standards	yes	The addition of “relations”, “individuals”, and “attributes” takes place in different ways.
	5. Avoid mistakes	no	
	6. Recognizing instead of remembering or recognizing before remembering	yes	Actions are not easy to find in jCORA 1.2.5.
	7. Flexibility and efficient use	no	
	8. Aesthetics and minimalist design	yes	Not all information relevant for use is available.
	9. Help with troubleshooting	yes	Error messages often prove to be incomprehensible.
	10. Help and documentation	yes	The help button has no function.
Other Heuristics	11. Perception control	no	
	12. Joy of Use	yes	The design of the CBR tool jCORA does not appear to be “contemporary”.

Table 56: Heuristics for problem identification for jCORA

In the second step, a usability test was carried out in the form of a field test. For this purpose, a survey tool was created that was answered by the test subjects. The results of the usability test show that there are usability problems with key functions of the CBR tool jCORA, particularly with regard to problem description and similarity calculation. The overall error rate was 47%. The results of the field test are shown in Table 57.

Usability component	Indicators	Results of the jCORA usability test (Average values)
Effectiveness	Completeness of task processing	92%
	Error rate	47%
Efficiency	Time of processing the tasks	9 min
Satisfaction	Reuse	80%
	Recommend to colleagues	60%
	Satisfaction	40%

Table 57: Results of the field test for jCORA

Another finding of the field test, which is not directly aimed at the usability problem of the CBR tool jCORA, is that there exists a fundamental need for such a tool, even if this particular one is rated as inadequate in terms of its usability.

The third step involved analyzing the CBR tool jCORA's usability problems. For this purpose, usability goals were defined that were to be achieved with regard to usability improvement by solving the already outlined usability problem. This means that the usability test for the click prototype should achieve at least the same values in terms of effectiveness, efficiency, and satisfaction as the usability test for jCORA. The aim was to achieve a significantly higher usability for the click prototype. For this purpose, the values in Table 57 should be exceeded. Furthermore, five user stories were formulated as part of the analysis, which were to form a basis for the design of a click prototype, as functionalities were formulated in these user stories that jCORA does not currently have

and which are to be added later in the click prototype. Table 58 below shows the user stories on which the analysis of jCORA's usability problems is based.

(End) user role	Aim	Reason
Senior Sales Manager	• Status of a project • My projects	• Faster recognition of the importance of a project • Faster adaptation of individual projects
Solution Manager	• Qualified retrieval of comparable content • Recording of own ratings	• Time saving • Standardization • Quality control
Technical Consultant	My expertise in intuitive handling and interpersonal understanding	Make customers' work easier
Client & Bid Manager	Reference comparisons	Easily find the right references that have already been prepared for any further tender.
Subproject manager & Business Analyst	Search for solutions that have solved the challenges in the data context (data migration, data maintenance, data conversion)	Identification of potential solutions for projects that have already been implemented

Table 58: User stories to analyze jCORA's usability problems

The usability goals and user stories from the analysis step were incorporated into the fourth step, in which several design proposals were developed based on common knowledge management tools. Further design decisions concerned aspects such as typography and symbols.

Taking into account the user stories, usability goals, and the design proposal, a click prototype with a total of 532 slides was developed using Adobe-XD software. Figure 103 below shows an example of a slide of the click prototype based on the project-related case specification; cf. WEBER et al. (2023), p. 81.

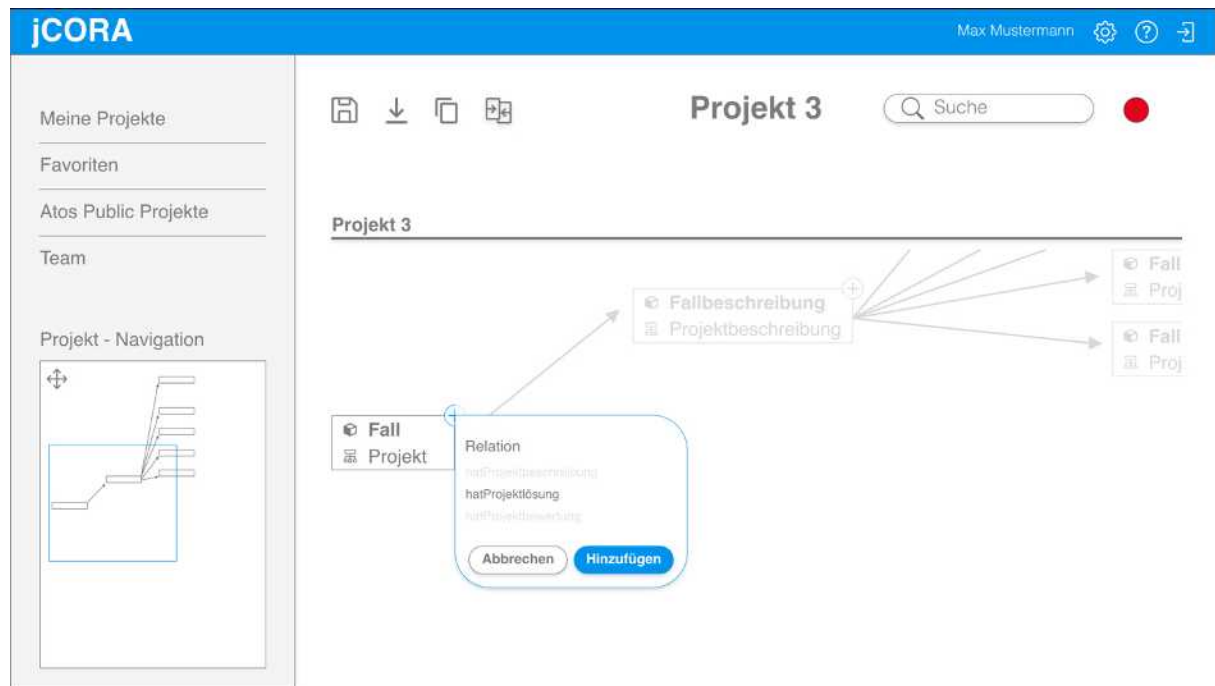


Figure 103: Click prototype

The click prototype was first subjected to a heuristic evaluation and then to a usability test, similarly to the previous procedure for the CBR tool jCORA. The heuristic evaluation showed that the new click prototype has only one usability error category—namely the often unfamiliar, technical terms for ontology components, such as “class”, “non-taxonomic relation”, “individual”, and “attribute”; see the heuristic “Match between system and real world” in Table 59 below. This table summarizes the results of the heuristic expert evaluations, comparing the results for the prototype CBR tool jCORA on the one hand and the newly designed click prototype on the other.

	Heuristics	Usability problem (yes/no) for the tool jCORA	Usability problem (yes/no) for the click prototype
Heuristics from NIELSEN	1. Visibility of the system status	no	no
	2. Correspondence between the system and the real world	yes	yes
	3. User control and freedom	no	no

	4. Consistency and standards	yes	no
	5. Avoid mistakes	no	no
	6. Recognizing instead of remembering or recognizing before remembering	yes	no
	7. Flexibility and efficient use	no	no
	8. Aesthetics and minimalist design	yes	no
	9. Help with troubleshooting	yes	no
	10. Help and documentation	yes	no
other Heu- ristics	11. Perception control	no	no
	12. Joy of use	yes	no

Table 59: Comparison of usability problems
between the CBR tool jCORA and the click prototype

The evaluation of the usability test for the new click prototype showed a significant improvement. The error rate was only 8% (in the first field test with jCORA it was 47%) and the recommendation rate for the application also increased, which further underlines the fundamental need for such an application. Table 60 below shows the comparison of the field test between the CBR tool jCORA and the click prototype as a revised user interface. Improvements in the indicators for usability components are highlighted in bold in the column for the click prototype.

Usability component	Indicators	Result of the usability test for jCORA (average values)	Result of the usability test for the click prototype (average values)
Effectiveness	Completeness of task processing	92%	96%
	Error rate	47%	8%
Efficiency	Time of processing the tasks	9 min	7 min
Satisfaction	Reuse	80%	80%
	Recommend to colleagues	60%	80%
	Satisfaction	40%	80%

Table 60: Comparison of the field tests
between the CBR tool jCORA and the click prototype

In short, the click prototype achieved a significant improvement in usability compared to the prototypical CBR tool jCORA. The knowledge gained from this provides a basis that can be used for the user-friendly design of the frontend for a future, professionally implemented ontology-supported CBR system as a cloud-native application.

However, it must also be mentioned that although the click prototype demonstrates a possible design of the user interface, and is helpful for demonstrating said interface and user interactions, it does not—and cannot—conclusively represent the frontend of an application. This is due to the fact that click prototypes are usually limited to the representation of user interface and interactions, but do not cover the concrete functionality required for the integration of a backend. In addition, click prototypes only provide a static representation of the user interface, while “real” applications require dynamic elements, real-time data processing, security aspects, and performance optimizations that only occur during implementation and may require adjustments to the user interface. For the above-mentioned reasons, problems can arise during the real implementation of

a frontend that require the user interface to be adapted and, if this is not the case, are at the expense of the heuristics considered above. Due to this lack of real implementation, it is therefore ultimately not possible to ensure that the user interface provided as a click prototype can be fully integrated into an ontology-supported CBR system as a cloud-native application.

4.3.2 Design of the backend

4.3.2.1 Preliminary considerations for backend design

The clients of the frontend of an ontology-supported CBR system access a backend via methods of an application programming interface (API). A client can be, for example, an application designed for mobile devices, an application designed exclusively for the web, or a specific client within an existing company-related application, such as SAP or Sharepoint. In the following, we explain the API methods provided by a Representational State Transfer Application Programming Interface (RESTful API). The RESTful API is an interface between IT systems (here between client and backend) that uses the Hypertext Transfer Protocol (HTTP) for its communication and transfers data using JSON (JavaScript Object Notation).

A backend can be divided into a database component and a middleware component.

The database component of the backend is irrelevant in the following consideration, as it is intended exclusively for storing the data and no further consideration is made here with regard to data storage. A standardized and free database can be provided in the cloud using a standard product such as DynamoDB. The security-critical IT project ontology can be stored in this database.

The middleware component of the backend consists of an API gateway and at least one serverless function. The API gateway serves as the “gateway” for all communication requests from the clients to the backend. The API gateway processes the communication requests, which are expected in a predefined API format and transported via the Hypertext Transfer Protocol (HTTP). The requests are forwarded by the API gateway to the underlying serverless functions. Figure 104 below shows an example of the serverless function `ermittleKonzeptAehnlichkeit` (serverless functions are indicated here by this special font). The associated resource path `/KonzeptAehnlichkeit` (resources and resource paths are identified by this special font), which leads to the call of the serverless function, is also shown. The responses from the serverless function represent the calculation results that are transmitted to the requested clients via the API

gateway. Figure 104 below illustrates the previously explained functionality of the middleware component of the backend of an ontology-based CBR system.

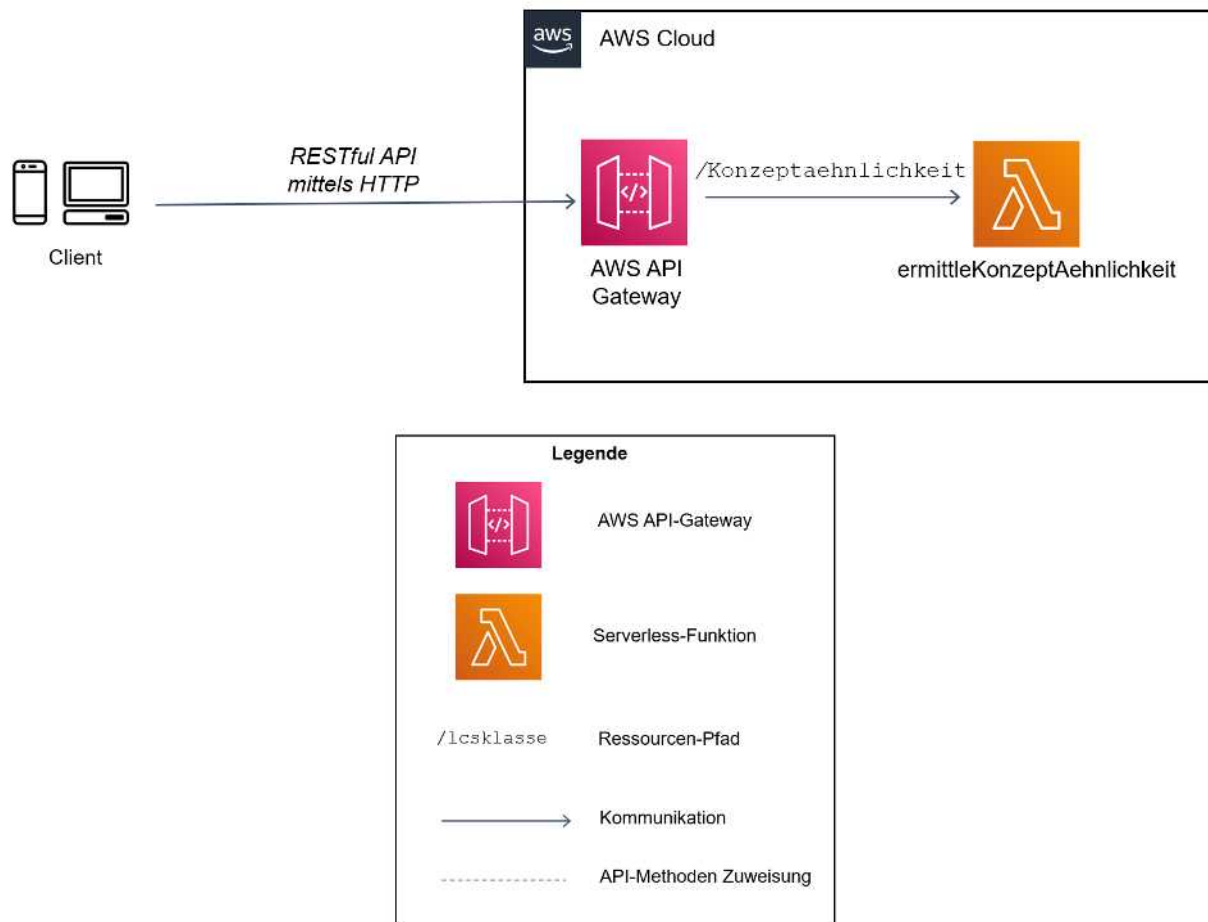


Figure 104: Representation of the middleware component of the backend of an ontology-based CBR system

The following explanations focus exclusively on the serverless functions that are provided in a cloud environment and map the business logic of a company that carries out its project management AI-based using ontologies and case-based reasoning.

An API gateway is part of an API management tool that mediates between a client and several serverless functions. It serves as a central interface that receives all API methods—such as PUT, ANY, GET, POST, PATCH, OPTIONS, HEAD, and DELETE—from the clients, forwards them to the required serverless functions, and returns the calculated results to the clients. An API gateway operates a series of APIs at a specific node, which can be called up using a Uniform Resource Locator (URL).

Figure 105 below shows an example of the “www.jcora.de” node, with the API methods GET, POST, ANY, and DELETE as examples. The API method GET is linked to the

serverless function `ermittleLCSKlasse` so that the serverless function `ermittleLCSKlasse` is executed when the GET method is called (using the resource path `www.jcora.de/lcsklasse`).

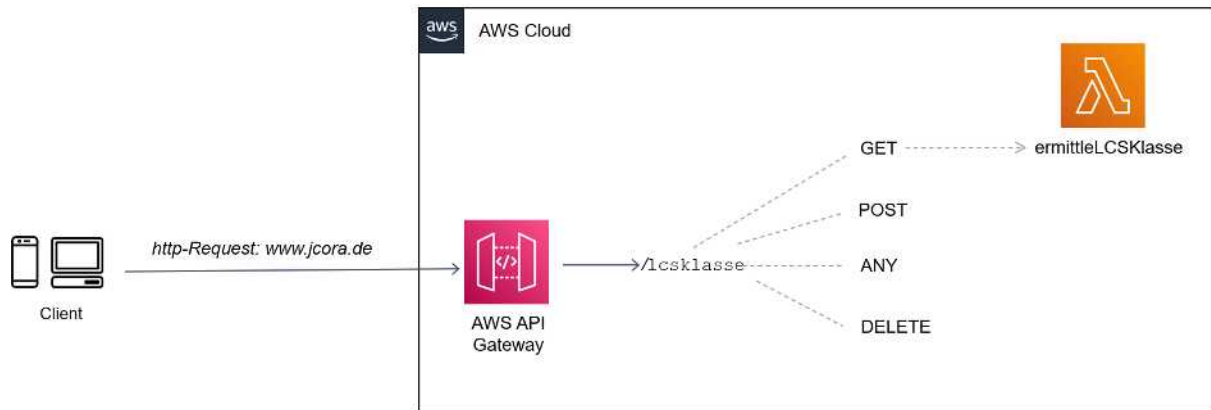


Figure 105: Access to the API of the resource `lcsklasse`

An API gateway supports two types of API methods, namely RESTful and Web Socket API. Only the RESTful API methods are relevant in this article. Therefore, API methods will always be understood as RESTful API methods in the following.

Figure 106 below shows an example of the configuration of the API gateway for the resource `lcsklasse` with the resource path `/lcsklasse`. Various API methods with corresponding serverless functions can be specified for this resource. In Figure 106, the GET method has been configured with the serverless function `ermittleLCSKlasse`. Each resource can have several API methods, but only of one method type. Therefore, the resource `lcsklasse` cannot have another GET method.

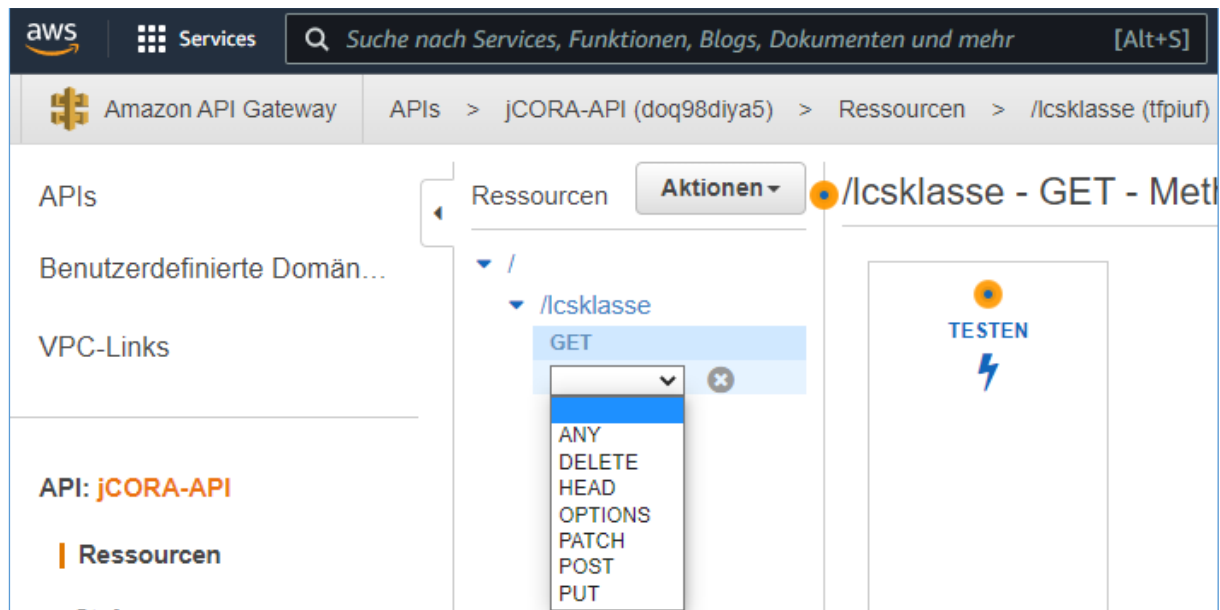


Figure 106: AWS API gateway configuration

The main task of the API gateway is to encapsulate the serverless functions for the clients. The advantages of using an API gateway are listed below.

- **Security:** By using an API gateway, not all functions need to be made available to the “outside world”. Serverless functions are not directly accessible to clients, but can only be accessed via the API gateway with prior authorization.
- **Network routing:** With direct communication between the client and the serverless functions, a transaction requires multiple function calls. This approach can lead to multiple network round trips between the client and the server, resulting in significantly higher latency.
- **Logging and monitoring:** The API gateway can serve as a centralized logging and monitoring entry point to monitor critical applications through a centralized entry point.
- **Independence of serverless functions from end devices:** As the serverless functions are accessed via the API gateway, there is no hard coupling between the frontend and backend. This enables flexible configuration of the clients.
- **Further development of an application:** If serverless functions are further developed, a direct call from a client can lead to errors in the client applications because there is a direct link between the client and the serverless function. However, API gateways can be used to adapt the serverless function using a consistent API call structure without causing an error, because the client and serverless function define what is to be passed and returned.

The use of a RESTful API method for an ontology-supported CBR system as a cloud-native application first requires the specification of the general structure of an HTTP request and an HTTP response.

- An HTTP request consists of a “request line” (which represents the URL call) and the “HTTP header fields”. A “message body” can optionally be present.
- An HTTP response consists of a “status line”, the “HTTP header fields”, and a “content type”.

Figure 107 below illustrates the facts explained above.

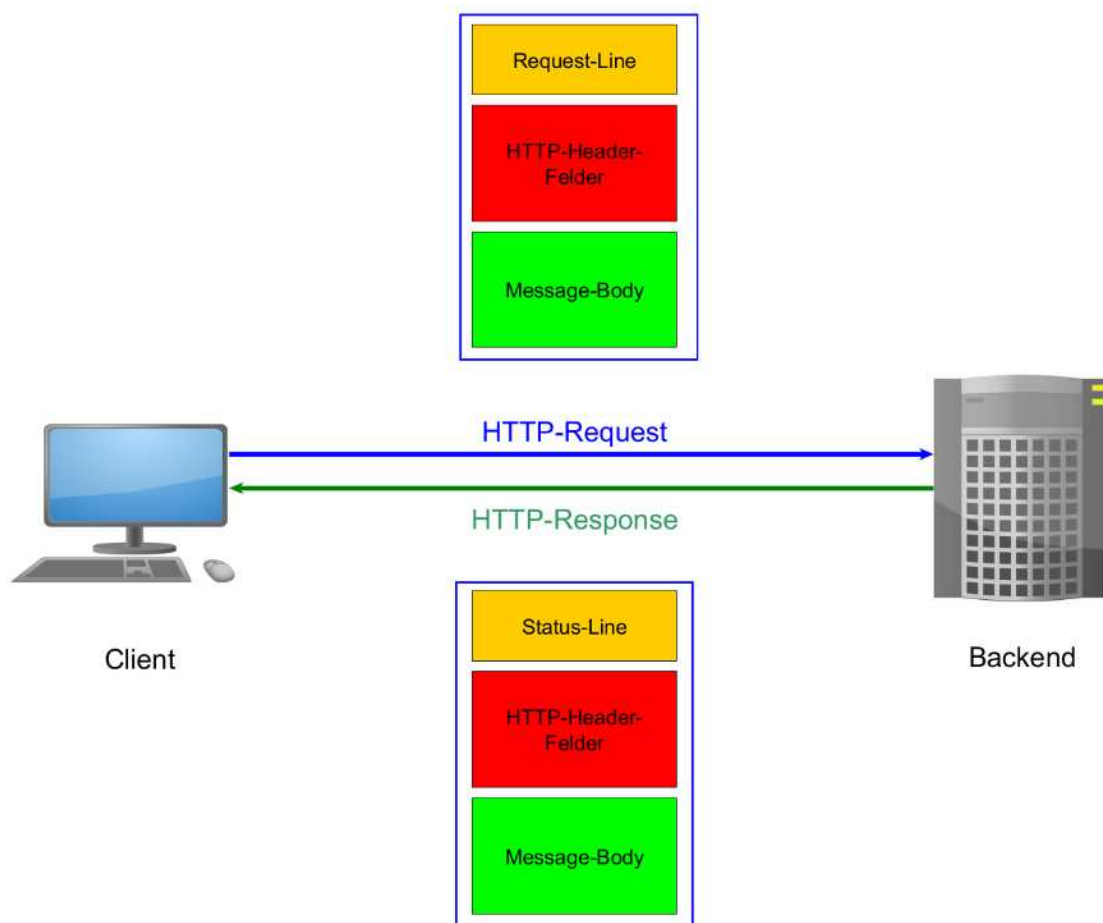


Figure 107: Client-backend communication via HTTP request and HTTP response

In the following, we will not further discuss the configuration of the API gateway, the RESTful APIs with the associated HTTP requests and HTTP responses, as well as the serialization of the calculation results of the serverless functions into the JSON format for the transfer between backend and frontend: Although these aspects are of considerable importance for the design of a backend for an ontology-based CBR system as a

cloud-native application, they are not the focus of the following investigations. This is justified as follows:

- The serverless functions map the business logic.
- The advantages of a cloud environment are anchored in the serverless functions.
- The necessary transfer parameters in the HTTP header can be derived from the transfer parameters of the serverless functions.
- The necessary return parameters in the HTTP response are also determined by the serverless functions.
- The serialization of the objects into a JSON format can be carried out using a standard function.
- Amazon Web Service offers a “configurative” procedure for the use of Amazon’s own API gateway in order to store expected function requirements (e.g., expected transfer parameters) and expected function responses for an implemented serverless function

The challenge in designing a backend for an ontology-supported CBR system as a cloud-native application lies mainly in the development of the serverless functions for mapping the business logic. The following explanations therefore relate exclusively to the development of such serverless functions.

4.3.2.2 Designing the middleware for the backend of a cloud-native application

The Python programming language is used to implement the serverless functions. Python is a “modern” object-oriented programming language; cf. DOWNEY (2021); KLEIN (2021); LUTZ (2007).

The choice of the Python programming language for implementing the serverless functions of an ontology-based CBR system is justified as follows:

- The Python programming language is currently the most widely used programming language for the development of backends; cf. CASS (2022); STACK OVERFLOW (2021). According to the study by POPULARITY OF PROGRAMMING LANGUAGE (2023), as of October 2022, Python is the most popular programming language for developers.

- Python's further development is driven by a large and active community. According to the study by SLASHDATA (2022), p. 13, as of the first quarter of 2022, Python has the largest backend developer community and the second largest developer community overall in terms of common programming languages.
- The Python programming language is distributed under an open source license and can be used free of charge. The intellectual property rights behind the Python programming language are held by the non-profit organization "Python Software Foundation"; cf. PYTHON (2023a). It manages the open source licensing for Python version 2.1 and higher; cf. PYTHON (2023a). The organization's mission statement states that the central Python distribution is made available to the entire public free of charge; cf. PYTHON (2002). This includes the Python programming language itself, its standard libraries and documentation, installation programs, source code, and training materials; cf. PYTHON (2002).
- Python is fully supported by all cloud environments and all examples and explanations are displayed in Python. This ensures a high degree of portability, as an application can be deployed in different cloud environments. The development examples from Amazon Web Service and Google Cloud Platform are used as examples; cf. AMAZON WEB SERVICES, INC. (2022a) and. GOOGLE DEVELOPERS (2022).
- Python is a programming language that promotes a concise and easy-to-understand programming style. As a result, Python-based applications are formulated much more concisely than those of other programming languages; cf. PYTHON (2023b); STEYER (2018), p. 3.
- Python is considered a common programming language for cloud-based developments, especially in the field of artificial intelligence and data analysis; cf. SLASHDATA (2022), p. 13. Various practice-oriented introductory literature on AI techniques—such as FROCHTE (2021), RASCHKA/MIRJALILI (2019), MÜLLER/GUIDO (2017), or RASHID (2017)—use Python as a programming language to explain AI techniques.
- Python offers the use of numerous free modules to extend functions. The OWL-ready2 module in particular represents a key advantage for using the Python programming language to implement an ontology-supported CBR system as a cloud-native application. The OWL-ready2 module offers methods for processing ontologies using Python.

- Python is considered one of the most secure programming languages. The MEND study (2018) examined the programming languages used in the last 10 years. According to this study, the programming language C occupies the “questionable” first place, with 47% of all reported vulnerabilities. Java is in third place, with 12% of all reported vulnerabilities. Python is in 5th place, with only 6%, followed by C++, also with 6%, and Ruby with 5%.

Based on the aforementioned advantages, the following exemplary design of the functionalities of an ontology-supported CBR system as a cloud-native application is carried out using the Python programming language.

An example of the implementation of a serverless function using Python is presented below. To simplify the language, we use the term “function” synonymously with the full term “serverless function”. All the functions programmed from this chapter onwards were both made available serverless and programmed using the Python programming language. The term “function” must be distinguished from the term “Python method”. Python methods are those offered by using modules, such as the Python method `startswith()`, which is offered by the Python module `String`. The Python module `String` from Python is a built-in module that does not need to be imported additionally.

The following source code for the function `pruefNUTSCode` has the variable `NUTSCode` as a transfer parameter. If the content of the `NUTSCode` variable begins with `DE`, the message “The area is in Germany” is returned. The check is carried out using the predefined Python method `startswith()`. The data type “String” is only assigned to the variable `NUTSCode` at runtime when this Python method is used. If the variable `NUTSCode` does not begin with `DE`, the following message is returned: “The area is not in Germany”. Single-line comments in Python begin with the `#` character.

```
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
```

Figure 108: Source code for the function `pruefNUTSCode`

Each phase of the CBR cycle—retrieve, reuse, revise, and retain—can be implemented as an independent function and provided in the cloud environment. The CBR cycle is represented by all of the functions, as shown in Figure 109 below.

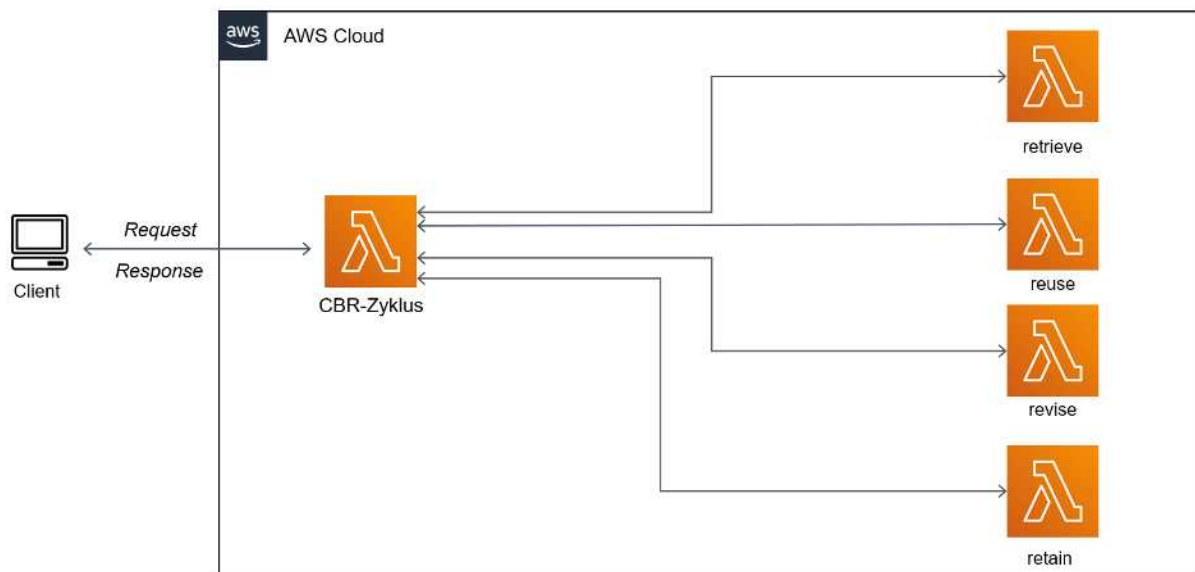


Figure 109: Functions for the CBR cycle

When the “CBR cycle” function is executed, the individual phases of the CBR cycle, which are implemented as independent functions, are called and executed. The respective results are returned to the higher-level “CBR cycle” function, which performs the further calculation steps on this basis.

Later on, we provide an example of the implementation of the retrieve phase with the similarity calculation to illustrate the conceptual design of the backend. The retrieve phase is of particular importance within the CBR cycle because the similarity algorithm causes a high computing load during it and represents a complex calculation. By dividing the similarity algorithm into different functions and outsourcing it to a cloud environment, the resources available for the individual functions can be scaled. Further subdivision allows the advantages of a cloud environment to be used for individual functions, e.g., to enable the scalability of individual parts of the calculation. Figures 110 to 112 below illustrate the interaction of the similarity functions, which together implement the similarity calculation for an ontology-based CBR system in the retrieve phase.

Figure 110 shows the function for calculating the complete similarity between two individuals (*isim*). It consists of the function for determining the class similarity (*ksim*)

and the function for determining the partial individual similarity (esim). The input parameters for the relevant function are shown above an arrow and the calculation results of the function (output parameters) are shown below an arrow in JSON format.

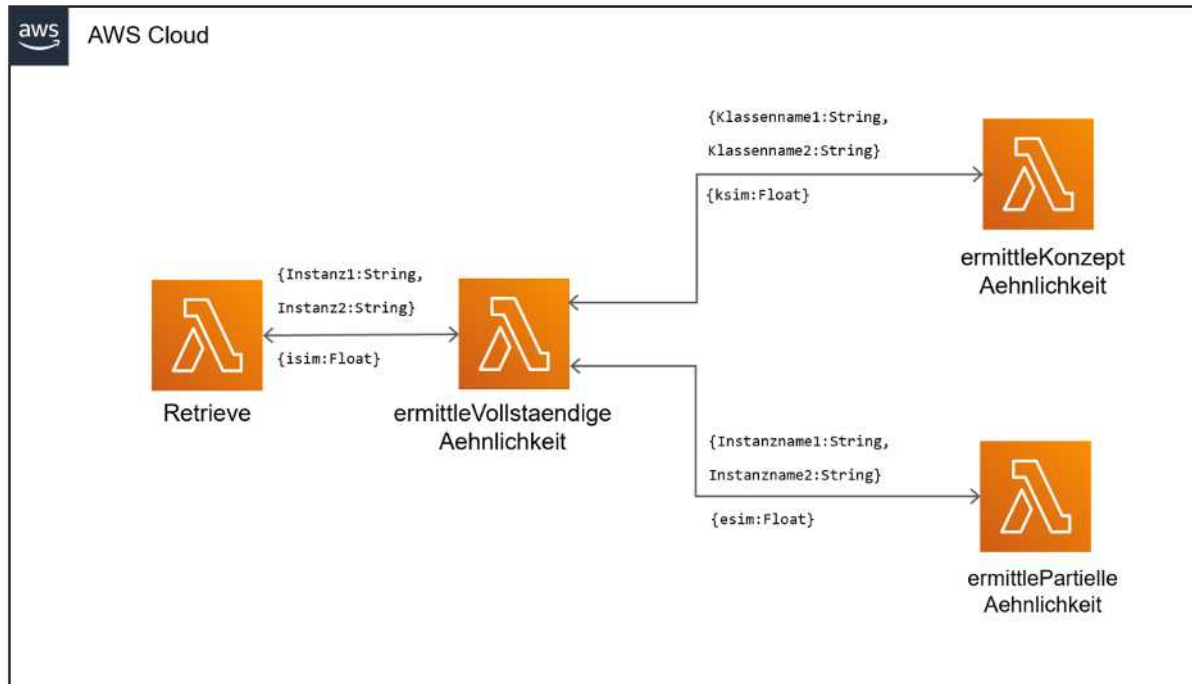


Figure 110: Functions for calculating similarity

Figure 111 below shows the further subdivision into functions for determining the class similarity, consisting of the function for calculating the semantic distance (**ermittle SemantischeDistanz**) and the function for determining the class properties of a class (**ermittleAehnlichkeitKonzepteigenschaft**).

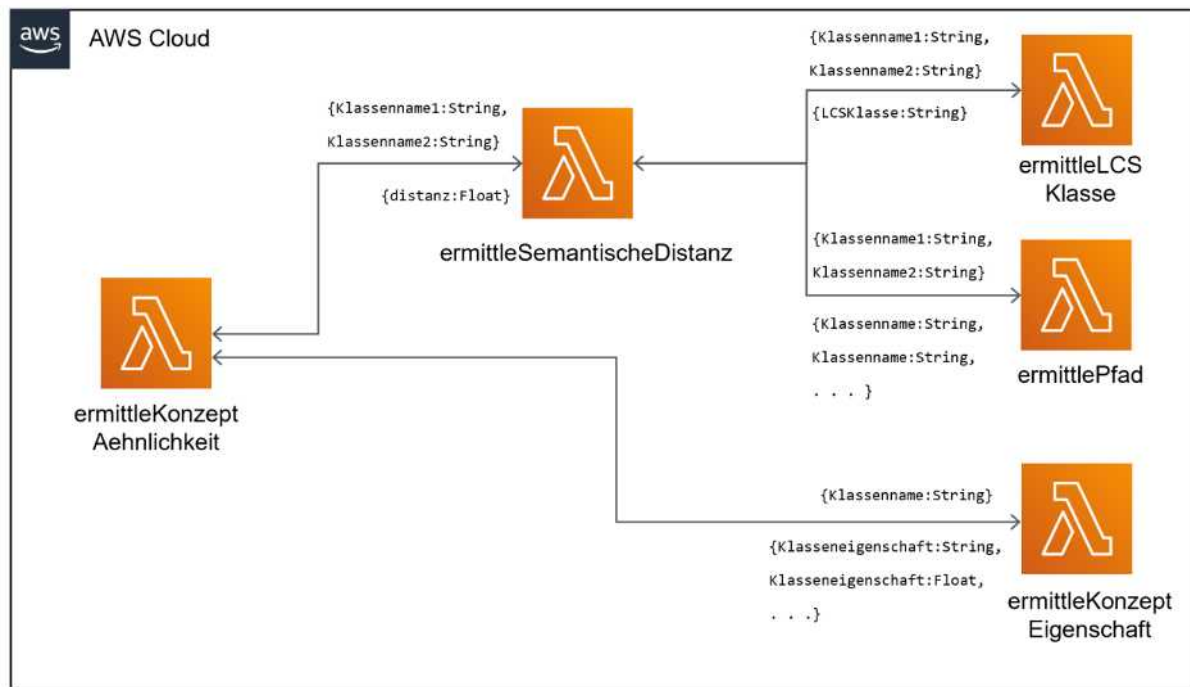


Figure 111: Functions for calculating class similarity

Figure 112 below shows the further subdivision into functions for determining the partial individual similarity. Three functions are implemented as examples. They calculate the similarity of individual properties in relation to three exemplary similarity types: CPV code, NUTS code, and string values. These functions for determining the similarity of individual properties are not only offered in the Amazon Web Service, but also on the Google Cloud Platform.

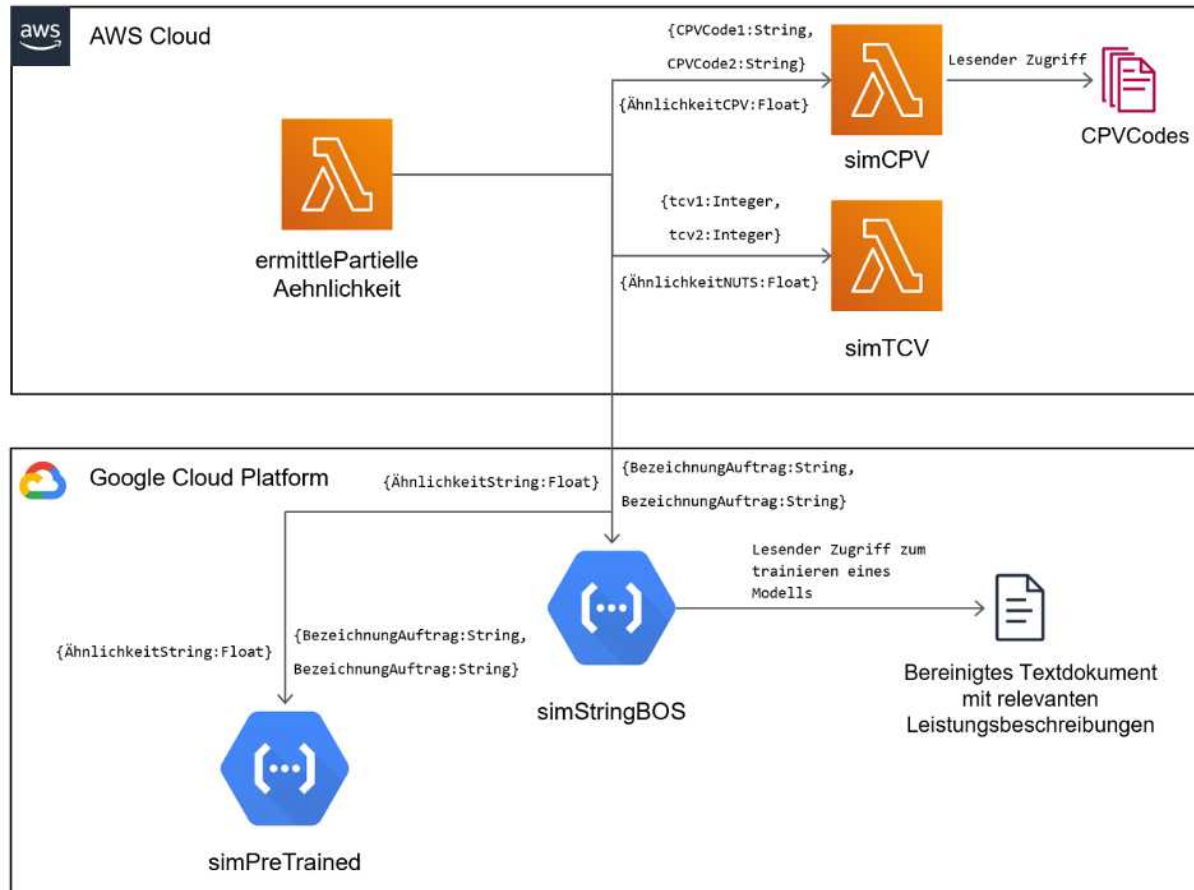


Figure 112: Functions for calculating partial individual similarity

We will explain the functions shown in Figures 110 to 112 in detail in a later chapter. Our focus here lies on the function **simStringBOS**. This function proves to be particularly interesting because it is based on the Word2Vec technology developed by Google and offers starting points for further possible uses—as we will discuss in detail later. The input parameters *BezeichnungAuftrag: String, BezeichnungAuftrag: String* specified in Figure 112 for the funktion **simStringBOS** are only selected as examples. This function can be used to compare different individual properties that are based on the “String” data type.

The methods or modules of a programming language are not always sufficient for processing an ontology. However, a module has existed in the Python programming language since 2017 that supports the construction, manipulation, and processing of ontologies. It is the “OWLReady2” module, which is currently available in version 2-0.39 (as of December 30th, 2022) and can be used free of charge under the GNU LGPL license. OWLready2 was developed in the research laboratory “*laboratoire d'informa-tique médicale et d'ingénierie des connaissances en e-Santé*” (LIMICS) of the University of Paris by LAMY; cf. LAMY (2023), p. 1. The module is used in various current

publications for processing ontologies; cf., e.g., DI MARTINO et al. (2022), p. 431; GUSKOV et al. (2022), p. 368; SARKER et al. (2021), p. 78. In STACKOVERFLOW—one of the best-known online communities for developers—there is a separate area for OWLReady2-related problems to get help with development problems; cf. STACK OVERFLOW (2023).

The *OWLReady2* module allows the object-oriented implementation of ontologies; cf. LAMY (2021), p. 5. The module also enables large ontologies to be loaded, as it can load ontologies that are several hundred gigabytes in size; cf. LAMY (2017), p. 23. In addition, access to ontology components is permitted by means of a special search method.

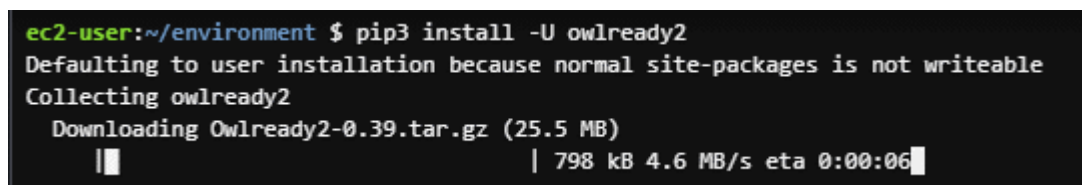
In order to be able to use the OWLReady2 module for programming with Python, it must first be installed. The module management program “pip3” is used for this purpose. It enables the automated downloading, installation, and updating of Python modules from the Python repository, called Python Package Index (PyPI), via the Internet. The shell command line for installing a Python module looks like this:

```
1. pip3 install -U name_des_moduls_dass_installiert_wer-  
den_soll
```

This management program can be executed in the shell command line on Unix/Mac or in the command prompt on Windows systems. The command line installs any Python module. If the module already exists, the module is updated. The OWLReady2 module is installed using the following command line.

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

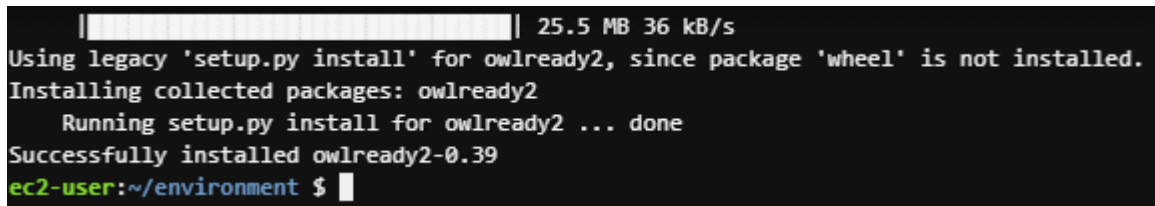
The command line can be executed directly in the integrated browser-based development environment ICloud9 from Amazon Web Service. Figure 113 below illustrates the download and installation process that runs automatically when the above command line is executed in the ICloud9 development environment of Amazon Web Service:



```
ec2-user:~/environment $ pip3 install -U owlready2  
Defaulting to user installation because normal site-packages is not writeable  
Collecting owlready2  
  Downloading Owlready2-0.39.tar.gz (25.5 MB)  
    | 798 kB 4.6 MB/s eta 0:00:06
```

Figure 113: Installing the OWLReady2 module

Once the installation is complete, a message is displayed stating that the OWLReady2 module has been successfully installed:



```

| 25.5 MB 36 kB/s
Using legacy 'setup.py install' for owlready2, since package 'wheel' is not installed.
Installing collected packages: owlready2
  Running setup.py install for owlready2 ... done
Successfully installed owlready2-0.39
ec2-user:~/environment $

```

Figure 114: Successful installation of the OWLReady2 module

After successful installation, the OWLready2 module can be used in the source code. There are two ways to import a module into the source code. The first way is to use the OWLReady2 methods directly. The import for direct use of the methods looks as follows:

```

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

```

The second way is by using the methods via the module designation OWLReady2:

```

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

```

In principle, both ways are possible. However, the first, direct method is generally preferred for the comprehensibility of the code. The documentation for OWLReady2 also recommends this type of import; cf. LAMY (2021), p. 83. We accordingly follow this recommendation in this article.

The module *Gensim* is an open-source Python module that provides several algorithms for analyzing large text documents; cf. ŘEHŮŘEK (2022). For example, the semantic structure of a document can be analyzed automatically using machine learning methods. The name “Gensim” is derived from “Generate Similar”. The module was developed in 2008 as a collection of Python scripts for the Czech Digital Mathematics Library (dml.cz) project; cf. ŘEHŮŘEK/SOJKA (2010), pp. 48–49; SOJKA (2009), in particular pp. 75–76. The Python scripts were used to create a short list of the most similar math articles to a given article; cf. ŘEHŮŘEK/SOJKA (2010), p. 47. ŘEHŮŘEK implemented an independent Python module on this basis and further developed the scripts as part of his dissertation; cf. ŘEHŮŘEK (2011), pp. 19–34, 37–66, and 67–78. The Gensim module is now considered a robust and frequently used module for the automatic semantic analysis of texts; cf. EL-AMIR/HAMDY (2020), p. 53; SARKAR (2019), p. 255. Gensim is used in a variety of projects; cf. for example NUGROHO et al. (2023), p. 294; PANDAY/SAHU (2023), pp. 294–295; TAYLOR/DU PREEZ (2023), pp. 539 and 550. Gensim is a public

project that is open for further development. The current Gensim version 4.3.0 (as of December 21st, 2022) was released in December 2022; cf. GITHUB (2022a).

The Gensim module provides various submodules and already has pre-trained models that can be used to calculate the similarity between terms. An overview of the pre-trained models can be found on GITHUB (2022b). We use only the Word2Vec submodule from Gensim in this article.

The term “training” is explained in more detail below. Gensim provides pre-trained models that have been pre-trained on the basis of the entire Wikipedia (as of 2017 in English), for example, and can be used for text analyses without the need for re-training. This saves the entire process of training text documents. Other models worth mentioning include Google’s pre-trained model, which was pre-trained on the basis of Google News (with a volume of 100 billion words). There also exist other pre-trained models for various languages, e.g., German, Chinese, and French.

In this paper, in addition to a self-trained model based on (previously anonymized) performance descriptions from safety-critical IT projects, we used a pre-trained model in order to be able to compare the different results. The larger models in particular, such as those from Google, require a high computing capacity (despite their pre-training) in order to be able to perform a similarity calculation during the runtime of the program execution. Although this capacity can be provided at runtime by a cloud environment, it incurs higher costs as more power is consumed. This is a non-negligible limitation when using large pre-trained models. We will return to this later.

It should also be noted that, strictly speaking, when using the Gensim module in the context of similarity calculation, it is not terms (in the semantic sense) but words (in the syntactic sense) that are meant. We will clarify this distinction later. However, for the sake of simplicity, the terms “term” and “word” are used synonymously in the following.

The Gensim module and its Word2Vec submodule can be imported on the Google Cloud Platform without prior installation. The module is imported using the following command:

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

Word2Vec is a word embedding technique published by MIKOLOV in 2013 as part of his work at Google, and which has since been publicly available to all users. MIKOLOV conducted the basic research into the vectorial representation of words while he was still working at Microsoft. They were published in MIKOLOV/YIH/ZWEIG (2013), which is

considered the central basis for Word2Vec. Other relevant works on this topic are MIKOLOV et al. (2013a) and MIKOLOV et al. (2013b).

The Word2Vec technique converts text into word vectors to capture the “semantics” of words and the relationship between them, and to calculate similarities between words on this basis. Word2Vec’s goal is to group vectors of similar words in a vector space in order to recognize the context; cf. MIKOLOV et al. (2013a), p. 2; MIKOLOV/YIH/ZWEIG (2013), p. 746.

The debate as to whether the Word2Vec technique can actually capture the “semantics” of words is controversial. Strictly speaking, Word2Vec merely captures syntactic coincidences of words in text corpora, learns statistical relationships between words, and uses them to derive the most probable (statistically speaking) text additions with new words in incomplete texts. This cannot be regarded as genuine “semantic” understanding, but is based on a “sophisticated” analysis of the quantitative-statistical—and therefore purely syntactic—correlations between words. This becomes clear, for example, from the challenges Word2Vec faces in distinguishing between several meanings of a word (polysemy) and in identifying different words that are spelled the same (homonymy). Both aspects can lead to incorrect “semantic” term representations in Word2Vec. Despite these “semantic doubts” and their well-founded debatability, we will continue to use the term “semantics” in the following with regard to the Word2Vec technique. This is justified by the fact that the Word2Vec technique can draw analogical conclusions and enables the creation of word vectors in which (syntactically) similar words are represented by similar vectors. Although this does not represent a comprehensive semantic realization of conceptual content in the semiotic sense, it does reflect the ability to model nuances of meaning and relationships between words in an abstract vector space. We will return to this later in more detail.

The Word2Vec technique uses a two-layer artificial neural network to calculate a model (this will be discussed in more detail later); cf. MIKOLOV/YIH/ZWEIG (2013), p. 746; MIKOLOV et al. (2013b), p. 1. Word2Vec increases its ability to recognize and output correlations through supervised learning. The representation of words by vectors makes it possible to recognize relationships through simple mathematical operations. An example often cited by MIKOLOV et al. (2013a), p. 2, and MIKOLOV/YIH/ZWEIG (2013), pp. 748–749 (similar also in MIKOLOV et al. (2013b), p. 1), is as follows:

$$\text{vec}(\text{“King”}) - \text{vec}(\text{“Man”}) + \text{vec}(\text{“Woman”}) \sim \text{vec}(\text{“Queen”})$$

The insight gained from such studies is that the modeling of word contexts using vectors results in syntactic and “semantic” (see above) relationships that can be analyzed using

mathematical operations; cf. MIKOLOV et al. (2013a), p. 2. The fundamentals of Word2Vec technology are crucial to recent advances in computer-based text processing; cf. DEVLIN et al. (2019), pp. 4171–4173; HOWARD/RUDER (2018), p. 328; RONG (2014), p. 1.

The Word2Vec technique offers two computational methods for learning word embeddings using an artificial neural network. The methods each calculate a model with which a center word can be predicted based on the adjacent context words or several context words can be output based on an input word. Figure 115 below shows the two models that can be calculated using the Word2Vec technique.

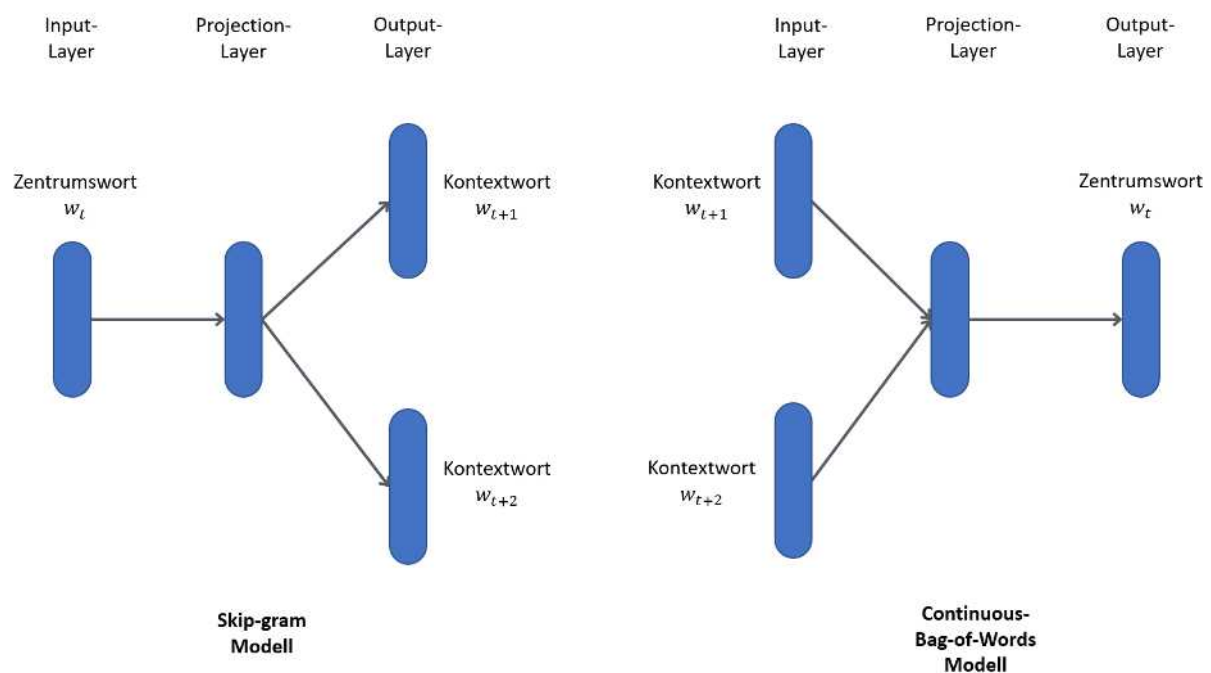


Figure 115: Word2Vec models

The two calculation methods provided by the Word2Vec technology are called:

- Continuous Bag of Words model (CBOW model)
- Skip-gram model (SG Model).

In the CBOW model, the searched word (center word) is predicted on the basis of adjacent context words; cf. MIKOLOV et al. (2013a), p. 5. The adjacent context words consist of words that are located before or after the center word being searched for. The order of the context words is not relevant in this model. The strength of the CBOW model lies

in its ability to capture syntactic relationships between two words better than the skip-gram model; cf. MIKOLOV et al. (2013a), p. 7.

In the skip-gram model, context words are predicted on the basis of an entered center word; cf. MIKOLOV et al. (2013a), p. 5. Basically, this model works in the opposite way to the CBOW model. The advantage of the skip-gram model is that it can recognize “semantic” relationships between two words better than the CBOW model; cf. MIKOLOV et al. (2013a), p. 7. Furthermore, according to MIKOLOV et al. (2013a), p. 9, the skip-gram model consumes fewer CPU resources than the CBOW model when training a model. We therefore use the skip-gram model as the basis for the following explanation of a similarity calculation.

The aim of the following is to explain the similarity calculation using the Word2-Vec technique on the basis of a skip-gram model using an example sentence. Some parameters (referred to below as hyperparameters), which are explained below, are important for the later implementation of the `simStringBOS` function. When using individual methods of the Word2Vec technique, various parameters must first be set. This requires a basic understanding of the calculations in the underlying artificial neural network.

The following explanations and calculations refer to the following example sentence from SENATSVERWALTUNG FÜR INNERES UND SPORT (2016), p. 2:

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

The example sentence also represents the text corpus under consideration here. This is simplistically assumed here, although a text corpus usually consists of an extensive text document and not just a single sentence. This text corpus (example sentence) Z comprises twelve words (represented as w_i , where i denotes the position in the text corpus), but only ten individual words, because the words “Berlin” and “die” occur twice in the text corpus. The individual words, which otherwise occur only once in the text corpus regardless of their spelling, are as follows:

1. “Die” und “die”
2. “Polizei”
3. “Berlin”
4. “und”
5. “Feuerwehr”
6. “betrieben”
7. “jeweils”

8. “unabhängig”
9. “voneinander”
10. “Leitstellen”

The individual words define the vocabulary V of the text corpus Z . The number of vocabulary elements is represented by $|V| = 10$. It should be noted that each word in the text corpus is counted once, regardless of its form (e.g., singular versus plural). This means that “control center” and “control centers” are considered two different words in the vocabulary. However, the skip-gram model generally has the advantage of being able to deal with such problematic cases in which different forms of a word occur. This is because it tries to identify semantically similar words. Therefore, it will “probably” learn that “Leitstelle” and “Leitstellen” occur in similar contexts and accordingly have similar word vectors. However, Word2Vec does not “inherently” recognize that these are the same term unless this is specifically taken into account during training, possibly through manual or semi-automatic preparation of the training data.

The text corpus is analyzed using a window (“window size”), which is specified with a fixed size m (e.g., $m = 1$). The window size with $m = 1$ does not represent a realistic application scenario. However, we chose this small window size as an example in order to reduce the complexity of the manual calculation and to be able to provide an explanation using a simple example. As a rule, larger window sizes are preferred in order to capture a broader context for word vectorization. A larger window m can lead to more training examples and to a higher accuracy of the similarity calculation, but at the expense of training time; cf. MIKOLOV et al. (2013b), p. 8. The choice of the “optimal” window size is a separate problem, which will be discussed later in this paper.

The word w_t in the middle of the window is referred to as the “center-word” or “target-word”. The preceding (w_{t-m}) and following (w_{t+m}) words are called “context words”. The window with the size m runs through the entire sentence to generate a word pair as a training pattern for the model. This word pair consists of a center word, which is later used as input, and one of its context words, which marks the target in the context. The context here means the context word that the model targets by determining the relationship and probability between the center word and the context word in a model.

Figures 116 to 118 below illustrate the first three iterations for extracting the training model. The center word w_t is shown in green, the context words w_{t-1} and w_{t+1} for the window with the size $m = 1$ are shown in gray.

1st iteration

die	polizei	berlin	und	die	feuerwehr	berlin	betreiben	jeweils	unabhängig	voneinander	leitstellen
-----	---------	--------	-----	-----	-----------	--------	-----------	---------	------------	-------------	-------------

Figure 116: First iteration

The training pattern is: *(die, polizei)*

2nd iteration

die	polizei	berlin	und	die	feuerwehr	berlin	betreiben	jeweils	unabhängig	voneinander	leitstellen
-----	---------	--------	-----	-----	-----------	--------	-----------	---------	------------	-------------	-------------

Figure 117: Second iteration

The training pattern is: *(polizei, die)*, *(polizei, berlin)*

3rd iteration

die	polizei	berlin	und	die	feuerwehr	berlin	betreiben	jeweils	unabhängig	voneinander	leitstellen
-----	---------	--------	-----	-----	-----------	--------	-----------	---------	------------	-------------	-------------

Figure 118: Third iteration

The training pattern is: *(berlin, polizei)*, *(berlin, und)*.

For the following explanations, we selected as an example the center word “feuerwehr” with the context words “die” and “berlin”. This represents the 6th iteration.

6th iteration

die	polizei	berlin	und	die	feuerwehr	berlin	betreiben	jeweils	unabhängig	voneinander	leitstellen
-----	---------	--------	-----	-----	-----------	--------	-----------	---------	------------	-------------	-------------

Figure 119: Sixth iteration

The training pattern is: *(feuerwehr, die)*, *(feuerwehr, berlin)*.

First, the center word and the context words are represented as one-hot coding in a vector. In one-hot coding, a feature is represented with a binary variable (1 or 0); cf. KULKARNI/SHIVANANDA (2021), p. 64; BISONG (2019), p. 336. One-hot coding makes machine processing possible. In the text corpus, the occurrence of the word is represented with a 1 and the non-occurrence with a 0. One-hot coding is often used in the field of machine learning; cf. KULKARNI/SHIVANANDA (2021), p. 64. Another limitation becomes clear here: With a large text corpus, a large vector is required to express each word as one-hot coding. This leads to a higher computing load, which can be covered by a cloud environment, but should not be neglected for large text corpora.

Table 61 below illustrates the one-hot coding for all words in the vocabulary.

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

Table 61: One-hot coding of the text corpus Z

In Table 61, each row represents a word from the vocabulary. The columns represent the different words in the vocabulary and indicate the position of each word in the vocabulary list. A “1” in a cell indicates that the word in this row is present in the relevant position in the vocabulary. A “0”, on the other hand, indicates that the word is not present in this row at the relevant position in the vocabulary.

Table 61 shows the vector of a word from the vocabulary as one-hot coding (shown in light blue). For example, the vector as one-hot coding for the word “polizei” is as follows:

$$\vec{polizei} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The dimension of the vector is 10×1 , as there are ten individual words in the vocabulary. The vectors of the center word and the context words—each in one-hot coding—are important for the following calculations, as they are included in the calculation as key components. The vectors of the center word “feuerwehr” (represented as $\vec{x}$), the context word “die” (represented as $\vec{y}_1$), and the context word “berlin” (represented as $\vec{y}_2$) are represented in one-hot encoding as follows:

$$\vec{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{y}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{y}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The vectors $\vec{y}_1$ and $\vec{y}_2$, which represent the context words, are also referred to as target vectors in the following calculation.

The vector $\vec{x}$ serves as input for the input layer of an artificial neural network. The artificial neural network is a feed-forward network. In this context, feed-forward means that within an artificial neural network, information is only forwarded forwards, i.e., to the next layer of neurons; cf. MATZKA (2021), p. 117. With regard to the Word2Vec technique, this means that information is only forwarded from the input layer to one or more hidden layers and then to the output layer. The procedure is explained in more detail below. In contrast to this are recurrent artificial neural networks, which are an extension of feed-forward, but in which information can also be passed to neurons in the same or previous layer; cf. MATZKA (2021), p. 128.

In the feed-forward network considered here, there is a hidden layer ($\vec{h}$) and an output layer ($\vec{u}$) in addition to the input layer ($\vec{x}$). The model of the Word2Vec technique essentially consists of the two weight matrices W_{Input} and W_{Output} for the weights of the connections between the neurons of the artificial neural network. The central task of the Word2Vec technique is to “optimize” the weight matrices W_{Input} and W_{Output} during the training process. In doing so, the model tries to maximize the probabilities for the occurrence of context words in relation to a center word in order to enable the best possible prediction of context words. Therefore, the model learns how the words in the

vocabulary relate to each other in “semantic”—strictly speaking, statistical (and therefore syntactic)—terms by adjusting the weights in the W_{Input} and W_{Output} matrices. This is explained in detail below.

Figure 120 below illustrates the calculation of the weight matrices W_{Input} and W_{Output} for the weights of the connections between the neurons of the artificial neural network. The following explanations are based on this figure.

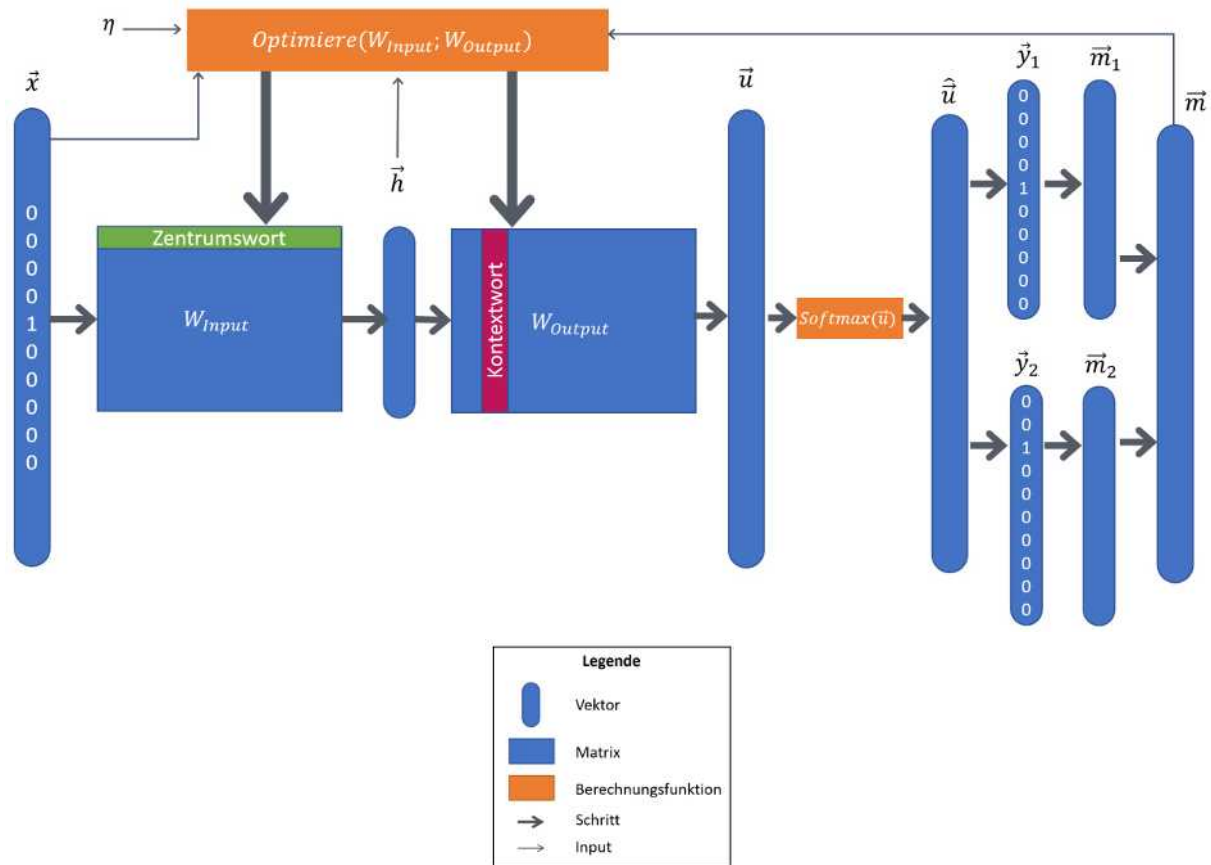


Figure 120: Calculation steps for the skip-gram mod

The variable N is defined in the next step. The variable N refers to the dimension of the word vector. The term “ N -dimensional word vector” is also used below. The dimension N of the word vector represents a hyperparameter for the hidden layer of the artificial neural network. In the Word2Vec technique, the hyperparameter N is referred to as “size” and is defined before the model is trained. A hyperparameter is a parameter in machine learning algorithms that is used to control the training algorithm and differs from other parameters in that it must be set before the model is trained; cf. AGRAWAL (2021), pp. 4–5. This also applies to the hyperparameters “window” m and “learning rate” η . We will discuss the learning rate η in more detail later on.

The weight matrices W_{Input} and W_{Output} contain the word vectors as rows or columns. Each row in W_{Input} and each column in W_{Output} corresponds to a word vector. The number of rows in W_{Input} and the number of columns in W_{Output} correspond to the size with $|V| = 10$ of the vocabulary V ; cf. RONG (2014), p. 8. See also Figure 120, where the row with the word vector for the center word is highlighted in green in the weight matrix W_{Input} , while the column with the word vector for the context word is highlighted in red in the weight matrix W_{Output} . At this point it becomes clear that although the weight matrices originate from the same vocabulary V , they represent different content. The weight matrix W_{Input} includes all center words, whereas the weight matrix W_{Output} includes all context words.

In the following example, we selected the dimension $N = 3$ for the N -dimensional word vector for the sake of simplicity. The N -dimensional word vector is much too small for a practical application, analogous to the window size with $m = 1$. In practical applications of the Word2Vec technique, the dimension of the word vector is usually $N = 300$; cf. MIKOLOV et al. (2013b), p. 6. However, we use the dimension $N = 3$ as a basis in order to avoid having to deal with excessively large weight matrices in the manual calculations.

The weight matrix W_{Input} has the dimension of 10×3 and is structured as follows:

$$W_{Input} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \\ w_{61} & w_{62} & w_{63} \\ w_{71} & w_{72} & w_{73} \\ w_{81} & w_{82} & w_{83} \\ w_{91} & w_{92} & w_{93} \\ w_{101} & w_{102} & w_{103} \end{bmatrix}$$

The weight matrix W_{Output} has the dimension of 3×10 and is structured as follows:

$$W_{Output} = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} & v_{17} & v_{18} & v_{19} & v_{110} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} & v_{26} & v_{27} & v_{28} & v_{29} & v_{210} \\ v_{31} & v_{32} & v_{33} & v_{34} & v_{35} & v_{36} & v_{37} & v_{38} & v_{39} & v_{310} \end{bmatrix}$$

The weight matrices W_{Input} and W_{Output} are initialized with random values at the beginning of the training process. This random initialization serves as a starting point for the training and enables the model to “optimize” the word vectors step by step; cf. AY-YADEVARA (2018), p. 170. The values in the weight matrices W_{Input} and W_{Output} are

machine-readable, but cannot be interpreted directly by humans. Machine-readable means that the values can be processed by computers.

In the following, the two weight matrices W_{Input} and W_{Output} are initially assigned any values:

$$W_{Input} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix}$$

$$W_{Output} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix}$$

In the next step, the vector $\vec{h}$ of the hidden layer is calculated. The hidden layer is also referred to as the “projection layer”, as the layer’s output is an N -dimensional word vector that is projected through the one-hot coding input vector $\vec{x}$.

In Figure 121 below, which shows an “adapted” section of Figure 120, illustrates the calculation of the vector $\vec{h}^T$. It should be noted that the vector $\vec{x}$ has the dimension 10×1 and the transposed vector $\vec{x}^T$ must be used for the following calculation so that a matrix multiplication with the matrix W_{Input} , which has a dimension of 10×3 , can be carried out. The transposed vector $\vec{x}^T$, which has a dimension of 10×1 , can be used to perform a matrix multiplication with the matrix W_{Input} , which has a dimension of 10×3 .

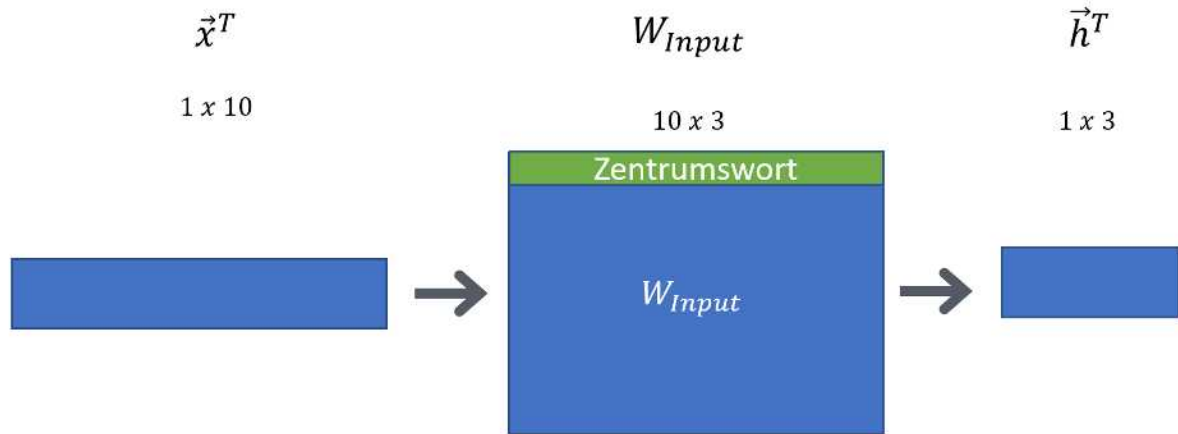


Figure 121: Calculation of the vector $\vec{h}^T$

The vector $\vec{h}^T$ is calculated as follows:

$$\vec{h}^T = \vec{x}^T * W_{Input} = [h_1 \quad h_2 \quad h_3]$$

$$\vec{h}^T = [0.13 \quad 0.14 \quad 0.15]$$

Before the prediction vector $\hat{u}$ can be calculated, the vector $\vec{u}$ must be calculated. Figure 122 below, which again shows an “adapted” section of Figure 120, illustrates the calculation of the vector $\vec{u}^T$:

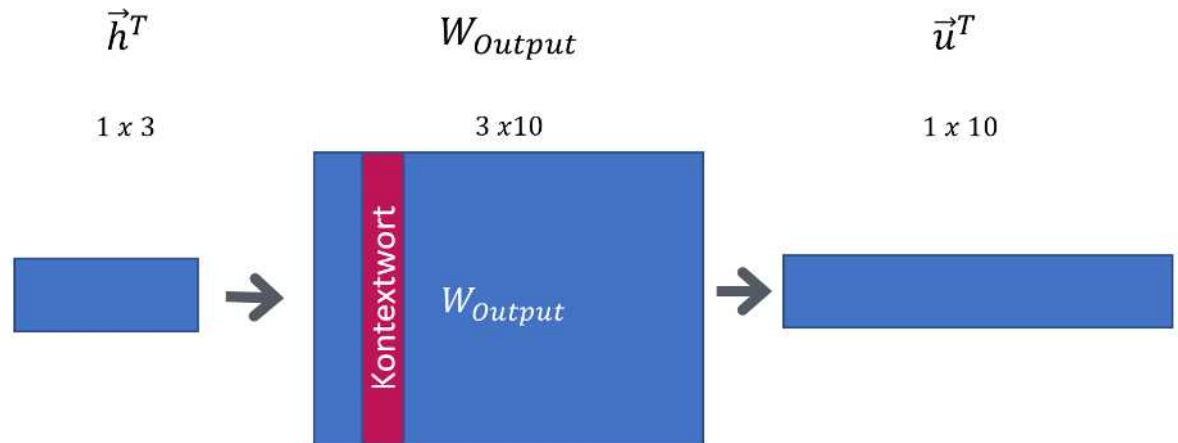


Figure 122: Calculation of the vector $\vec{u}^T$

The vector $\vec{u}^T$ is calculated as follows:

$$\vec{u}^T = \vec{h}^T * W_{Output} = [u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6 \quad u_7 \quad u_8 \quad u_9 \quad u_{10}]$$

$$\vec{u}^T = [0.06 \quad 0.076 \quad 0.092 \quad 0.108 \quad 0.124 \quad 0.139 \quad 0.155 \quad 0.171 \quad 0.187 \quad 0.203]$$

The vector $\vec{u}^T$ is used to calculate the prediction vector $\widehat{\vec{u}}^T$. To generate a probability distribution over all words for the vector $\vec{u}^T$, a softmax function σ is used, which normalizes the vector $\vec{u}^T$ to values between 0 and 1; cf. GOLDBERG/LEVY (2014), p. 2; RONG (2014), p. 8. “1” stands for the “certain result”, i.e., a probability of 100%, and “0” for the “impossible event”, i.e., a probability of 0%. The softmax function σ converts a k -dimensional real vector into a k -dimensional real vector whose components lie in the interval $[0,1]$ and add up to 1. The advantage of this function is that the input values can be any values from $\mathbb{R}$, but are mapped by the softmax function to the interval $[0,1]$ with the component sum 1, so that they can be interpreted as probabilities. If one of the input values is small or negative (large), the softmax function converts it into a small (large) probability. The softmax function is often used in machine learning in an artificial neural network; cf. PAAB/HECKER (2020), pp. 60–62.

Figure 123, below, which again shows an “adapted” section of Figure 120, illustrates the calculation of the prediction vector $\widehat{\vec{u}}^T$.



Figure 123: Calculation of the prediction vector $\widehat{\vec{u}}^T$

The prediction vector $\widehat{\vec{u}}^T$ is calculated as follows:

$$\widehat{\vec{u}}^T = \sigma(\vec{u}^T) = \left[\frac{e^{u_i}}{\sum_{j=1}^R e^{u_j}} \quad i = 1, \dots, R \right]$$

$$= \left[\frac{e^{u_1}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_2}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_3}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_4}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_5}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_6}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_7}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_8}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_9}}{\sum_{j=1}^{10} e^{u_j}} \quad \frac{e^{u_{10}}}{\sum_{j=1}^{10} e^{u_j}} \right]$$

Here u_i stands for the i -th element of the vector $\vec{u}^T$. R is the number of elements in the vector $\vec{u}^T$ and is here $R = 10$. The symbol Σ represents the sum of all elements with the running index j with $j=1, \dots, R$.

The prediction vector $\widehat{\vec{u}}^T$ has the following rounded (symbol: “ $\approx$ ”) values:

$$\widehat{\vec{u}}^T \approx [0.093 \quad 0.094 \quad 0.096 \quad 0.098 \quad 0.099 \quad 0.101 \quad 0.102 \quad 0.104 \quad 0.106 \quad 0.107]$$

In the next step, the error vectors for the two context words are calculated. Since the two vectors for the context words $\vec{y}_1$ (“die”) and $\vec{y}_2$ (“berlin”) result from the training pattern, the error vectors $\vec{m}_1$ and $\vec{m}_2$ are calculated by subtracting the prediction vector $\widehat{u}^T$ from the vectors $\vec{y}_1$ and $\vec{y}_2$ for the two context words; cf. GOLDBERG/LEVY (2014), p. 8. The advantage of using the window size m here becomes clear: It reduces the complexity of the calculation because the model only has to calculate the prediction errors for the context words within the specified window size m , for which $m = 1$ applies here.

A non-zero error value in the error vector indicates that the model’s predictions deviate from the actual observations, in this case the two vectors $\vec{y}_1$ and $\vec{y}_2$ for the two context words. The ideal value for the error vector would be $\vec{0}$. This would indicate that the model makes “perfect” predictions and that there is no deviation between the predictions and the actual context words. In practice, however, it is rarely possible to achieve an error vector of exactly $\vec{0}$, as deviations almost always exist. Therefore, the main goal of *optimization* by the error vector is to *minimize* this deviation as much as possible in order to *maximize* the prediction accuracy of the entire model. This defines a precise and at the same time operational optimization criterion for the calculations using the Word2Vec technique in the artificial neural network under consideration.

Due to the different dimensions between the prediction vector $\widehat{u}^T$ (dimension 1×10) and the vectors $\vec{y}_1$ and $\vec{y}_2$ (dimensions 10×1 in each case), the prediction vector is transposed. Figure 124 below, which again shows an “adapted” section of Figure 120, illustrates the calculation of the error vectors $\vec{m}_1$ and $\vec{m}_2$.

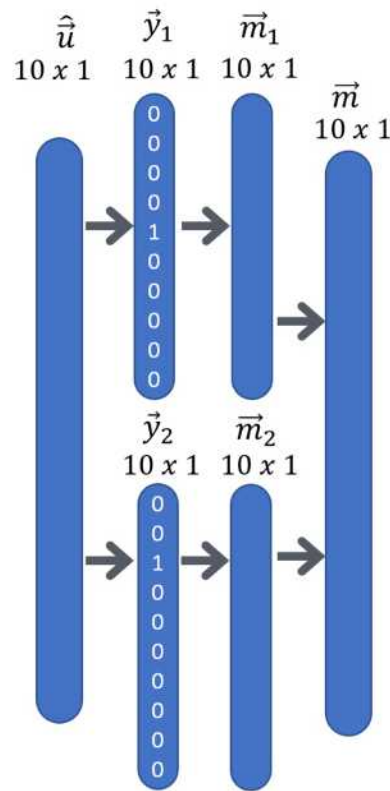


Figure 124: Calculation of the error vectors

The error vectors $\vec{m}_1$ and $\vec{m}_2$ are calculated as follows:

$$\begin{aligned} \vec{m}_1 &= \hat{u} - \vec{y}_1 & \vec{m}_2 &= \hat{u} - \vec{y}_2 \\ \vec{m}_1 &= \begin{bmatrix} -0.907 \\ 0.094 \\ 0.096 \\ 0.098 \\ 0.099 \\ 0.101 \\ 0.102 \\ 0.104 \\ 0.106 \\ 0.107 \end{bmatrix} & \vec{m}_2 &= \begin{bmatrix} 0.093 \\ 0.094 \\ -0.904 \\ 0.098 \\ 0.099 \\ 0.101 \\ 0.102 \\ 0.104 \\ 0.106 \\ 0.107 \end{bmatrix} \end{aligned}$$

The values of the prediction errors from the error vectors $\vec{m}_1$ and $\vec{m}_2$ are summed up for both context words “die” and “berlin” and displayed as the (aggregated) error vector $\vec{m}$:

$$\vec{m} = \sum_{c=1}^c \vec{m}_c$$

Here C stands for the number of error vectors for context words. In this case, the two error vectors are $\vec{m}_1$ and $\vec{m}_2$ (with $C = 2$), which are added together:

$$\vec{m} = \begin{bmatrix} -0.814 \\ 0.188 \\ -0.808 \\ 0.196 \\ 0.198 \\ 0.202 \\ 0.204 \\ 0.208 \\ 0.212 \\ 0.214 \end{bmatrix}$$

Once the error vector $\vec{m}$ has been calculated, the two weight matrices W_{Input} and W_{Output} are updated in order to “optimize” the model. This process is referred to as “back-propagation” and is based on the gradient descent method; cf. RONG (2014), pp. 17–20. The idea behind this is to gradually adjust the values in W_{Input} and W_{Output} so that the error vector $\vec{m}$ is minimized.

The first step is to explain how to update the weight matrix W_{Input} , which is referred to below as $W_{Input_{neu}}$.

To do this, the error vector $\vec{m}$ is multiplied by the transposed vector $\vec{h}$ from the hidden layer:

$$\vec{m} * \vec{h}^T$$

The updated weight matrix $W_{Input_{neu}}$ is calculated by multiplying the preceding product by the learning rate η and subtracting it from the old weight matrix $W_{Input_{alt}}$:

$$W_{Input_{neu}} = W_{Input_{alt}} - \eta * \vec{m} * \vec{h}^T$$

The learning rate η is once again a hyperparameter. It controls how quickly the artificial neural network model is adapted to the prediction problem under consideration. The learning rate determines the strength of the weight changes in the weight matrices W_{Output} and W_{Input} . This method is called the delta rule, or the WIDROW-HOFF method. Smaller learning rates require more training cycles given the smaller changes that are made to the weights with each update, while larger learning rates lead to faster changes and require fewer training cycles; see MIRFENDRESKI (2022), p. 143. A training cycle comprises one iteration over the entire training data set. As a rule, the learning rates

range between 1 and 0 in the field of machine learning; cf. AICHELE (2021), p. 11; KLÜVER/KLÜVER (2021), p. 14; KLÜVER/KLÜVER/SCHMIDT (2021), p. 189. For the Word2Vec technique, a learning rate between 0.025 and 0.0001 was used, which is considered by DI GENNARO/BUONANNO/PALMIERI (2021), p. 12328, as the starting point for determining an optimal learning rate.

The calculation with the aforementioned values is performed by first recalculating (“optimizing”) the weight matrix $W_{Input_{alt}}$ using the learning rate $\eta = 0.025$:

$$W_{Input_{neu}} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 * \begin{bmatrix} -0.814 \\ 0.188 \\ -0.808 \\ 0.196 \\ 0.198 \\ 0.202 \\ 0.204 \\ 0.208 \\ 0.212 \\ 0.214 \end{bmatrix} * [0.13 \quad 0.14 \quad 0.15]$$

The calculation result of $\vec{m} * \vec{h}^T$ is shown in the following step:

$$W_{Input_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 * \begin{bmatrix} -0.10582 & -0.11396 & -0.1221 \\ 0.02444 & 0.02632 & 0.0282 \\ -0.10504 & -0.11312 & -0.1212 \\ 0.02548 & 0.02744 & 0.0294 \\ 0.02574 & 0.02772 & 0.0297 \\ 0.02626 & 0.02828 & 0.0303 \\ 0.02652 & 0.02856 & 0.0306 \\ 0.02704 & 0.02912 & 0.0312 \\ 0.02756 & 0.02968 & 0.0318 \\ 0.02782 & 0.02996 & 0.0321 \end{bmatrix}$$

In the following step, the calculation result is displayed with regard to the multiplication with the learning rate:

$$W_{Input_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 0.1 & 0.11 & 0.12 \\ 0.13 & 0.14 & 0.15 \\ 0.16 & 0.17 & 0.18 \\ 0.19 & 0.20 & 0.21 \\ 0.22 & 0.23 & 0.24 \\ 0.25 & 0.26 & 0.27 \\ 0.28 & 0.29 & 0.3 \end{bmatrix} - \begin{bmatrix} -0.00265 & -0.00285 & -0.00305 \\ 0.00061 & 0.00066 & 0.00071 \\ -0.00263 & -0.00283 & -0.00303 \\ 0.00064 & 0.00069 & 0.00074 \\ 0.00064 & 0.00069 & 0.00074 \\ 0.00066 & 0.00071 & 0.00076 \\ 0.00066 & 0.00071 & 0.00077 \\ 0.00068 & 0.00073 & 0.00078 \\ 0.00069 & 0.00074 & 0.0008 \\ 0.0007 & 0.00075 & 0.0008 \end{bmatrix}$$

Finally, the two matrices mentioned above are subtracted. The weight $W_{Input_{neu}}$ looks as follows (and is further “optimized” in the following iterations):

$$W_{Input_{neu}} \approx \begin{bmatrix} 0.1026 & 0.2028 & 0.3030 \\ 0.3993 & 0.4993 & 0.5992 \\ 0.7026 & 0.8028 & 0.9030 \\ 0.0993 & 0.1093 & 0.1193 \\ 0.1294 & 0.1393 & 0.1493 \\ 0.1593 & 0.1692 & 0.1792 \\ 0.1893 & 0.1992 & 0.2092 \\ 0.2193 & 0.2293 & 0.2392 \\ 0.2493 & 0.2592 & 0.2692 \\ 0.2793 & 0.2892 & 0.2992 \end{bmatrix}$$

The weight matrix $W_{Output_{neu}}$ is calculated in a similar way to the weight matrix $W_{Input_{neu}}$. However, it should be noted that the weight matrix $W_{Output_{neu}}$, like the weight matrix $W_{Output_{alt}}$, is a (3×10) matrix. Therefore, the transposed vectors must be used to ensure a mathematically admissible calculation. This means that the vector $\vec{h}^T$ and the error vector $\vec{m}$ must be transposed before multiplication to ensure that the multiplication of both vectors forms a matrix with the same form as $W_{Output_{alt}}$:

$$\vec{h} * \vec{m}^T$$

The formula for calculating $W_{Output_{neu}}$ is as follows:

$$W_{Output_{neu}} = W_{Output_{alt}} - \eta * \vec{h} * \vec{m}^T$$

The calculation of $W_{Output_{neu}}$ with the aforementioned values is carried out as follows:

$$W_{Output_{neu}} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 * \begin{bmatrix} 0.13 \\ 0.14 \\ 0.15 \end{bmatrix} * [-0.814 \quad 0.188 \quad -0.808 \quad 0.196 \quad 0.198 \quad 0.202 \quad 0.204 \quad 0.208 \quad 0.212 \quad 0.214]$$

The calculation result of $\vec{h} * \vec{m}^T$ is shown in the following step:

$$W_{Output_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix} - 0.025 * \begin{bmatrix} -0.10582 & 0.02444 & -0.10504 & 0.02548 & 0.02574 & 0.02626 & 0.02652 & 0.02704 & 0.02756 & 0.02782 \\ -0.11396 & 0.02632 & -0.11312 & 0.02744 & 0.02772 & 0.02828 & 0.02856 & 0.02912 & 0.02968 & 0.02996 \\ -0.1221 & 0.0282 & -0.1212 & 0.0294 & 0.0297 & 0.0303 & 0.0306 & 0.0312 & 0.0318 & 0.0321 \end{bmatrix}$$

The calculation result of the multiplication with the learning rate is displayed in the following step:

$$W_{Output_{neu}} \approx \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \end{bmatrix} - \begin{bmatrix} -0.00265 & 0.00061 & -0.00263 & 0.00064 & 0.00064 & 0.00066 & 0.00066 & 0.00068 & 0.00069 & 0.0007 \\ -0.00285 & 0.00066 & -0.00283 & 0.00069 & 0.00069 & 0.00071 & 0.00071 & 0.00073 & 0.00074 & 0.00075 \\ -0.00305 & 0.00071 & -0.00303 & 0.000743 & 0.00074 & 0.00076 & 0.00077 & 0.00078 & 0.0008 & 0.0008 \end{bmatrix}$$

Finally, the two previous matrices are subtracted. The weight matrix $W_{Output_{neu}}$, which is further “optimized” in the following iterations, looks as follows:

$$W_{Output_{neu}} \approx \begin{bmatrix} 0.10265 & 0.19939 & 0.30363 & 0.39936 & 0.49936 & 0.59934 & 0.69934 & 0.79932 & 0.89931 & 0.99305 \\ 0.11284 & 0.11934 & 0.13283 & 0.13931 & 0.14931 & 0.15929 & 0.16929 & 0.17927 & 0.18926 & 0.19925 \\ 0.21305 & 0.2193 & 0.23303 & 0.23927 & 0.24925 & 0.25924 & 0.26924 & 0.27922 & 0.28921 & 0.29920 \end{bmatrix}$$

When the update of the two weight matrices W_{Output} and W_{Input} is complete, the current iteration is ended. The window of size m moves to the next center word (in this case the word “berlin”). The next iteration begins.

In summary, the skip-gram model attempts to “optimize” the values of the weight matrices W_{Output} and W_{Input} using the error vector $\vec{m}$ in such a way that the context words are predicted with maximum probability for a given center word. Maximizing the probability of predicting context words leads to the optimization of the weight matrices W_{Output} and W_{Input} . Essentially, the Word2Vec technique is used to learn statistical—often referred to as “semantic” (we will come back to this)—relationships between words by developing and “optimizing” vector representations for words from an underlying text corpus.

As details on the information technology implementation of the similarity algorithm for an ontology-supported CBR system in a cloud environment will probably only be of interest to IT and AI experts, we do not go into them in this article. Instead, detailed implementation explanations can be found in SETHUPATHY (2024), pp. 470–535.

5 Critical reflections

5.1 Ontology-supported case-based reasoning for the reuse of experience-based knowledge about safety-critical IT projects

The construction of the safety-critical IT project ontology depends on a large number of design decisions. Although almost every decision in the context of the construction of the safety-critical IT project ontology can be characterized as worthy of discussion, the following remarks focus only on a few construction decisions that we consider particularly prominent and valuable.

Firstly, it can be critically noted that we used as our basis a still incomplete PM domain ontology from the KI-LiveS project. The PM domain ontology has neither versioning nor citable documentation. In addition, we used OWLReady2 to determine that the PM domain ontology was incorrect, as duplicate name designations produced an error message.

The construction of the safety-critical IT project ontology was essentially limited to extending the PM domain ontology to include the aspects of safety-critical IT projects. Although we simplified the integration into the CBR tool jCORA by using the PM domain ontology, said ontology contains a large number of classes and properties that are not relevant for the safety-critical IT project ontology. The restrictions made mainly relate to the extended classes, relations, and attributes, so that it can be critically noted that for a completely integer application of the constructed ontology, the existing entities of the existing PM domain ontology should also have been provided with restrictions. This criticism also applies to the PRINCE2 and risk management ontology used. Furthermore, for the safety-critical IT project ontology, it cannot be expected that all linguistic means of expression for representing domain-specific knowledge are included.

The choice of the Protégé software tool should also be critically discussed. The errors recognized by OWLReady2 regarding duplicate names were not recognized by Protégé. Errors also occasionally occurred in the construction of the ontology with Protégé, notably the following:

- When constructing complex situations (e.g., inheritance), an error in Protégé can lead to the editor window being called by a different class than the one previously selected by the user.
- When installing plug-ins, it can occasionally happen that plug-ins cannot be used and have to be reinstalled.

- Protégé can no longer be used due to overlapping graphical elements; this often occurs when Protégé is restarted.
- Both the HermiT and the Pellet Reasoner of Protégé did not recognize the errors of double naming.

The use of the web-based ontology editor WebProtégé could presumably resolve these highlighted problems with Protégé. In particular, the high degree of updating of WebProtégé could be an indication that it has a significantly shorter error correction time than the desktop version of Protégé. If the safety-critical IT project ontology is further developed, it could be set up exclusively with the WebProtégé ontology editor, for example, in order to utilize the advantages of Internet-based application software. However, the user must be aware that Web Protégé has a more limited range of functions than desktop Protégé.

The ontology gains expressive power through the cardinalities and SWRL rules. However, the CBR tool jCORa cannot process SWRL rules. Nevertheless, we have decided in favor of constructing such rules in order to make them as expressive as possible in the sense of an independent ontology. A further limitation of SWRL rules is their use in an operational environment, which requires extensive maintenance.

Furthermore, one can criticize the “subjective” selection and small number of cases in the case base. On the one hand, the selection of cases ensures that the ontology provides the linguistic means of expression necessary for modeling the cases. On the other hand, we arguably did not check whether the constructed ontology also proves to be practicable in the context of other conceivable cases. The small number of cases in the case base does not allow any statement to be made about the runtime, e.g., for calculating similarity values and for determining the most similar projects. It can be recognized that the performance requirement for calculating the similarity of cases is increasing. However, it can be assumed that there are far more cases in the case base in the operational environment than in the ontology-supported CBR system considered here as a prototype, so that the runtime can increase significantly. An evaluation of the runtime based on a realistic number of cases in the case base was not carried out. It is also important to determine the number of cases above which the CBR system delivers “better” results in terms of identifying cases that are as similar as possible.

It can also be argued that not every class in the ontology of safety-critical IT projects could be constructed separately due to confidentiality levels and possible security concerns. In some places, there is also a lack of concrete reference to practice-relevant service descriptions due to possible security concerns. Although it is possible to derive the specific characteristics of safety-critical requirements via individuals, it was not possible

to adequately express further-reaching safety properties that are subject to a confidentiality level in the safety-critical IT project ontology.

Another point of criticism is the collection of terms for the class construction, which we carried out by means of an informal enquiry via email. It is conceivable that further terms could have been mentioned that would have been relevant for the construction, but were overlooked due to this choice of method. However, we deliberately chose this informal procedure so as not to rely exclusively on feedback from experts, but also to be able to use individual terms from the terms of reference for which there were no security concerns. In addition, we intended the informal enquiry to prevent us from influencing the answers to the terms and competency questions, while making it easier to document the response.

In addition, the adaptation of the solution of the most similar old project, which fulfills the required minimum similarity, to the description of a new project for the application of ontology-supported case-based reasoning to safety-critical IT projects is missing, so that strictly speaking there is no complete application of case-based reasoning. We do explain conceptually how the adaptation could take place (in particular the acquisition of adaptation knowledge). However, concrete adaptation rules for the domain of safety-critical IT projects and exemplary applications of these adaptation rules to safety-critical IT projects are missing. Nor did we consider the revise and retain phases. This is reserved for later research work—including by the second author of this article as part of his dissertation.

We could also have tested the safety-critical IT project ontology for its expressiveness in the operational environment, and incorporated user feedback in the context of possible tests into improving the ontology, among other things. It would be conceivable to use a possible procedure as in WEBER et al. (2023), pp. 38–56, where the researchers evaluated a user interface of the CBR tool jCORA via a user test in the form of a survey.

Finally, the selection of the CBR tool jCORA can be criticized. We could have selected this CBR tool on the basis of a catalogue of requirements from potential operational users of case-based reasoning, for example by applying the Analytic Hierarchy Process as a business management multi-criteria evaluation technique, which is used, for example, by BEIBEL (2011), pp. 49–132, in particular pp. 85–132, to evaluate as both an ontology editor and a CBR tool.

5.2 Cloud-native application for an ontology-supported case-based reasoning system

The design of an ontology-supported CBR system as a cloud-native application and the prototypical implementation of individual functions based on it depend on numerous design decisions, which we critically reflect upon in the following.

The first decision concerns the choice of cloud environment. We could have selected Microsoft Azure Cloud from Microsoft as both the primary and secondary platform. In the case of development in the Microsoft Azure Cloud, Microsoft's Visual Studio Online development environment would have to be used instead of the Cloud9 development environment used primarily. Visual Studio Online is particularly recommended for the development of Microsoft-related applications. The Google Cloud Platform, which we used as a secondary platform, could also have been used as the primary platform. In particular, the browser-based development environment Google Colab is designed for development with Python for machine learning, big data analyses, and all AI algorithms. As a cloud-hosted version of Jupyter Notebook, Colab offers free access to computer infrastructure such as storage, memory, processing capacity, graphics processing units (GPUs), and tensor processing units (TPUs) for AI algorithms. In this article, we developed the `simStringBOS` function based on Word2Vec with the help of Google Colab. It would be conceivable to implement all functions via Colab in the form of a Jupyter notebook and make them available exclusively in Google's cloud environment.

Another limitation is the lack of a database in the prototype implementation. The ontology is stored on a file server in the cloud environment in order to demonstrate the feasibility of an ontology-supported CBR system as a cloud-native application as a prototype. In an operational environment, however, the ontology would have to be stored in a database and protected by access rights. We did not do so in the prototype implementation of this article, nor did we sufficiently consider the entire role and rights management. Although the API gateway creates centralized access, this only represents a starting point for role and rights management. Other components, such as authentication mechanisms, would have to be considered subsequently. In principle, the manipulation of an online technology should be linked to a role, such as administrator.

The area of data protection comprises further concerns, which is particularly salient because security-critical IT projects involve projects with highly sensitive data that are generally subject to classification. In terms of data protection compliance, two areas need to be improved when designing an ontology-based CBR system as a cloud-native application.

Firstly, the choice of cloud provider is a point the literature has critically discussed from a data protection perspective; cf., e.g., KNEUPER (2021), p. 11; JÄGER/RIEKEN/ERNST (2020), pp. 3–10; VOIGT/VON DEM BUSSCHE (2018), pp. 315–317. It must be ensured that the German data protection regulations are complied with and that no unauthorized data access (e.g., by other foreign security authorities) is possible.

Secondly, in addition to legal data protection, technical data protection should also be taken into account in the form of additional functionalities. For example, in the concept presented here we have not sufficiently taken into account the aforementioned rights management and the pseudonymization and encryption of personal data. The use of the API gateway and the implementation using serverless functions offer various starting points for integrating technical data protection. One such starting point could be to encrypt the communication between the client, the API gateway, and the serverless functions using secure transmission protocols such as HTTPS. We did not consider such aspects here, deliberately accepting this limitation as our focus was on demonstrating the general feasibility of an ontology-supported CBR system for safety-critical IT projects.

Another fundamental point that plays a vital role alongside data protection is a cloud environment's security. Data can be stolen in a cloud environment through hacker attacks, which can pose a serious threat to security-critical IT projects with a certain confidentiality level. Furthermore, recent cases show that even cloud environments are not protected against attacks and that such attacks can lead to entire systems being deleted as a result. One such example is the recent case at CloudNordic, where a hacker attack caused considerable damage by encrypting all the data in the cloud environment and rendering it unusable; cf. KUHN (2023).

The small number of specific similarity functions implemented should also be critically scrutinized. The dictionary includes only four similarity types with assigned similarity functions; moreover, they are hard coded. For use in an operational environment, the dictionary means additional effort, as the assignment and maintenance must be carried out manually. Reliable automatic determination of the similarity types would be desirable, but will hardly be possible in practice.

The specific similarity function `simStringBOS` represents a central similarity function that calculates a similarity between two words from the area of safety-critical IT projects using Word2Vec. It is crucial that various critical aspects be taken into account when implementing this function, which must be carefully evaluated. Firstly, the fundamentally debatable interpretation of the similarity of Word2Vec must be criticized. In mathematical terms, similarity is usually defined as a measure of proximity or correlation between two objects. In Word2Vec, however, similarity is calculated on the basis of the

spatial proximity of the vectors in vector space. Words that occur in similar contexts and have similar meanings are close to each other in the vector space. It is assumed that words that occur in similar contexts also have similar meanings. Therefore, the spatial proximity of the vectors is used as a measure of the similarity between the words. It is important to note that the similarity calculated with Word2Vec is based on patterns (center words and context words) that the model has learned from the text corpus. There may be situations where the similarity does not always match the intuitive meaning or human understanding of similarity. The results strongly depend on the quality and representativeness of the text corpus. It should also be noted that Word2Vec, as a statistical method, is based on probabilities and distributions. Consequently, strictly speaking, the term “similarity” cannot always be interpreted in the classical mathematical sense, making it more appropriate to use the term “probability”.

In addition to the debatable interpretation of the similarity of Word2Vec, some decisions regarding the implementation of the `simStringBOS` similarity function also need to be scrutinized. We will first discuss the selection and anonymization of the performance descriptions. Although our training basis was made up of three performance descriptions that together comprise 1,300 pages, this training basis could be expanded and broadened to include further performance descriptions. As already described in the previous explanations, the challenge is to provide training principles that are relevant to practice on the one hand, but also to remove critical information or documents with confidentiality levels from the training principles on the other. In order to meet this challenge, we limited the anonymization and selection to three service descriptions.

The data preparation of the text document in the `simStringBOS` function can also be discussed in pre-processing. In particular, chapter numbers and special characters such as “(” are not removed. Although automated removal is possible, it would result in numbers such as “110” and “112”, which are particularly important in the area of safety-critical IT projects, also being removed. The definition of exceptions and manual editing of the text document would be conceivable. The data preparation carried out is sufficient to illustrate the feasibility, as it provides correct calculation results for the similarity of two words. Overall, however, it should be noted that said preparation can be improved by further data cleansing. The extent to which the text basis is manipulated by excessive data preparation and the original content is too strongly altered must be weighed up.

There also exist new technologies such as the Transformer approach (Transformer-based Language Models and Large Language Models) to determine the similarity of words. A transformer is a type of artificial neural network that was first introduced by VASWANI et al. in 2017. The essential core of a transformer is its ability to process in-

formation (e.g., from natural language texts) in parallel and thus quickly train the underlying artificial neural network; cf. VASWANI et al. (2017), p. 2. The algorithm uses multi-head attention mechanisms to understand the meaning of each word in the context of its environment; cf. VASWANI et al. (2017), p. 1. Although both Word2Vec and Transformer are based on artificial neural networks, there are some important differences. Word2Vec is based on a relatively simple artificial neural network that learns word embeddings. A transformer, on the other hand, has a more complex, multi-layered architecture for artificial neural networks, which is based on the “attention mechanism”. The attentional mechanism is a concept of machine learning that was published by LUONG/PHAM/MANNING. It is used to give more weight to certain parts of an input than others by calculating a weighting for each element; cf. LUONG/PHAM/MANNING (2015), p. 1. When processing natural language texts, the attention mechanism is used as part of the transformer to understand the meaning (frequency of coincidences) of a word in the context of its environment; see LUONG/PHAM/MANNING (2015), pp. 3–4. The transformer calculates a weighting for each word in the text and then only takes into account the words with the highest weightings when processing the text. This enables the transformer to understand the meaning (frequency of coincidences) of a word in the context of its environment; cf. LUONG/PHAM/MANNING (2015), p. 8. Word2Vec, on the other hand, processes texts sequentially and only takes into account word relationships in a limited context (window size), while the transformer enables parallel processing and understands the meaning of each word in the context of its environment. The transformer approach has proven to be very successful in processing natural language texts. A well-known example is ChatGPT. The performance of the transformer approach is higher than that of Word2Vec because the method enables parallel processing. To summarize, the transformer approach offers a more complex model architecture for artificial neural networks compared to Word2Vec, which can lead to faster training and better results in the calculation of word similarities and is therefore an interesting option for use in a CBR system. Current literature considers the use of a transformer promising. See for example NIGHOJKAR/KHLYZOVA/LICATO (2022), p. 6; TABINDA KOKAB/ASGHAR/NAZ (2022), p. 11; MOHD/DHASMANA/UPADHYAY (2021), p. 2399. The sources mentioned also make a comparison between a Transformer and a Word2Vec approach.

When implementing the `simStringBOS` function, the selection of parameter values must be considered critically. Although we selected and tested various values for the parameters in this article, further parameter combinations can be presented that go beyond our present scope.

Due to their prototypical nature, the implemented functions can be criticized in that the following points of software development were not sufficiently taken into account:

- No error handling has been provided to intercept any incorrect entries made by a user. This can be the case, for example, if a function expects a string data type for the similarity calculation, but the user enters an integer data type.
- The source code is not uploaded to a software repository and versioned.
- No automated software tests were carried out.

Although we based our design of the user interface as a click prototype on the publication WEBER et al. (2023), the design decisions made there can be discussed. The usability engineering lifecycle according to NIELSEN was chosen for the design of the user interface. This can be criticized from two perspectives. Firstly, the usability engineering lifecycle according to MAYHEW is more detailed than the usability engineering lifecycle according to NIELSEN. Secondly, the aspects of NIELSEN could have been combined with those of MAYHEW. Another point of criticism regarding the design of the user interface is that the accessibility of the click prototype was given too little consideration in the design decisions. Accessibility is becoming increasingly important for applications in the business environment. This is reflected both in the Barrier-free Information Technology Ordinance (BITV) and in ISO standard 9241 (Ergonomics of Human-System Interaction). A further limitation of the user interface is that the click prototype was only designed in the form of a web design and mobile applications were not taken into account. For further critically reflected design decisions, which are not discussed further here, please refer to WEBER et al. (2023), pp. 103–104.

The selection of the similarity algorithm for the exemplary implementation of the concept of an ontology-based CBR system as a cloud-native application can be viewed critically. The selection of application components is based on subjective criteria, such as the expected complexity of the implementation, the necessary automatic processing of an ontology and the potentially high resource consumption for the similarity calculation. Furthermore, it cannot be ruled out that a complete implementation as a cloud-native application could lead to problems elsewhere, for example when configuring the RESTful API structure for the interaction between the frontend and backend or when providing the user interface in a cloud environment. Strictly speaking, this article does not offer a complete concept that considers all areas (frontend and backend) in interaction; instead, the individual areas are considered separately and the focus is on the implementation of the similarity algorithm.

Finally, the cloud-native application as a whole must be critically evaluated. An error in one function does not remain isolated, but can affect the entire similarity algorithm. The more functions are implemented and interact with each other, the more important testing

and monitoring them becomes. Although the functions' complexity is reduced by isolated functions, the resulting distributed system across cloud providers increases the complexity of the backend. Instead of a monolith, several functions must be monitored in parallel.

6 Conclusions on the scientific findings obtained

We began this article by presenting scientific problems that were to be solved by means of the following intended scientific results:

- A safety-critical IT project ontology
- A CBR system with integrated ontology for safety-critical IT projects
- Safety-critical IT projects as cases in a CBR system
- A similarity calculation in the CBR system
- A similarity algorithm as serverless functions in a cloud environment
- Similarity functions as serverless functions in a cloud environment, including specific similarity functions for processing qualitative information from safety-critical IT projects using artificial neural networks

In the following, we draw our conclusions about the above-mentioned, intended scientific results in order to check whether this contribution has actually achieved them.

For the construction of a safety-critical IT project ontology, we developed a procedure based on NOY/MCGUINNESS, which we extended by an additional activity—the definition of rules. In order to construct a practice-oriented safety-critical IT project ontology, we used competency questions and terms formulated by experts and terms from relevant service descriptions, taking into account the confidentiality classification. We then constructed the safety-critical IT project ontology using the Protégé ontology editor, applying the PM domain ontology and the PRINCE2 and risk management ontology as a basis. The intended scientific result of a safety-critical IT project ontology can be considered to have been achieved, taking into account the limitations described in chapter 5.1.

The CBR tool used for the construction of ontology-supported CBR systems was jCORA, which supports heterogeneous case bases and has an integrated similarity algorithm. The implementation of the safety-critical IT project ontology in such a CBR system represented the second intended scientific result. It can be regarded as fulfilled because the ontology has been successfully integrated into the CBR tool jCORA for case construction and similarity calculation.

In order to achieve the intended scientific result of capturing safety-critical IT projects as cases in an ontology-supported CBR system, we specified such cases on the basis of the safety-critical IT project ontology. The intended scientific result can be regarded as

fulfilled, as we specified three cases for safety-critical IT projects in an exemplary manner.

We carried out a similarity calculation with the specified cases in the ontology-based CBR system. However, we cannot say with certainty whether this calculation was carried out correctly. We identified limitations, such as a lack of application performance and incorrect similarity calculations when using global individuals. It should also be critically noted that we did not weight the individual properties of the cases separately, instead uniformly giving them a weight of 1.0, although different weightings may be required in practice. However, we chose this approach merely to demonstrate the basic feasibility of the concept presented here for ontology-based CBR systems. Furthermore, specific similarity functions are missing, so that the universal similarity functions may be insufficient and lead to incorrect calculations. We therefore consider the intended scientific result of the similarity calculation in the CBR system to be only partially achieved.

In order to achieve the intended scientific result of providing the similarity algorithm as a serverless function to demonstrate the feasibility of ontology-supported case-based reasoning as a cloud-native application, we selected Python as the programming language and Amazon Web Services (AWS) and the Google Cloud Platform (GCP) as the cloud environments. We used the Python module OWLReady2 to process the ontologies. When we were initially importing the ontologies into the cloud environment with OWLReady2, we found that the underlying PM domain ontology had inconsistencies that made this action impossible. The ontology editor Protégé had not reported these inconsistencies. After manually correcting the inconsistencies in the ontology (duplicate names), we managed to perform an import with OWLReady2. Both OWLReady2 and Protégé use HermiT as a reasoner, but Protégé did not show the inconsistencies.

Another challenge was the implementation of the similarity algorithm, which we did as a serverless function. We found that the calculated similarity between two individuals was perceived as too low despite correct application of the algorithm. To counteract this impression, we used not only common class properties but also class properties of the same similarity type for comparison. Although the algorithm was hard-coded in some areas, in particular through the use of a dictionary, the implementation shows in principle that the similarity algorithm can be provided by serverless functions and also offers starting points for further development. Since the complete similarity algorithm was implemented as a serverless function, which led to a correctly calculated similarity, the intended scientific result can be regarded as achieved.

The final intended scientific result of this paper is the implementation of specific similarity functions for the processing of qualitative information from safety-critical IT projects with artificial neural networks. We implemented specific similarity functions based on similarity tables (coded in the `simTCV` function as an example), data types such as Boolean, and two specific similarity functions based on Word2Vec models. For demonstration purposes, we prepared a text corpus of service descriptions for safety-critical IT projects, and trained it in two models. The specific similarity function is able to calculate the similarity of two words from the service descriptions. The challenge in calculating the model lies in the parameterization. Predefined models can help here, but are domain-independent. Even though current developments such as ChatGPT show that Word2Vec models could be replaced by transformer approaches in the future, there are currently no implementations to use such algorithms for ontology-supported case-based reasoning. Since the specific similarity functions were implemented as serverless functions, the intended scientific result can be considered achieved.

The intended scientific results are summarized with the scientific results actually achieved using “Harvey balls”:

Harvey Ball	Description
	No result
	Result only achieved in small parts
	Result partially achieved
	Result largely achieved
	Result achieved in full

Table 62: Harvey Ball definitions for the qualitative comparison of the intended scientific results

The comparative result is shown in Table 63 below. For more detailed explanations, cf. SETHUPATHY (2024), pp. 552–553.

Intended scientific result	Scientific result actually achieved	Degree of fulfillment
Safety-critical IT project ontology	Safety-critical IT project ontology	●
CBR system with integrated ontology for safety-critical IT projects	CBR system with integrated ontology for safety-critical IT projects	●
Safety-critical IT projects in the form of cases in a CBR system	Safety-critical IT projects in the form of cases in a CBR system	●
Similarity calculation in the CBR system	Similarity calculation in the CBR system with universal similarity functions	◐
Similarity algorithm as server-less functions on a cloud environment	Similarity algorithm as server-less functions on a cloud environment	●
Similarity functions as server-less functions in a cloud environment including two specific similarity functions for processing qualitative information from safety-critical IT projects using artificial neural networks.	Similarity functions as server-less functions in a cloud environment including two specific similarity functions for processing qualitative information from safety-critical IT projects using artificial neural networks.	●

Table 63: Comparison between intended and actually achieved scientific results

7 Outlook for further research needs

The potential for further development already indicated in this article offers a number of interesting questions that still need to be investigated.

The first starting point for in-depth investigations is the further development of the safety-critical IT project ontology to include additional linguistic means of expression and domain-specific rules. The construction of further SWRL rules would increase the safety-critical IT project ontology's informative value and could also serve as a basis for the development of specific adaptation rules. It would be conceivable to gain further linguistic means of expression by involving additional experts and further performance descriptions and to provide linguistic means of expression from other project management methods, such as PMI or Scrum, and integrate them into the safety-critical IT project ontology. For example, it would be desirable to implement adaptation rules that enable an exchange between project management methods, so that a project carried out with PRINCE2 receives suggestions for new projects with regard to an adaptation to project management methods, such as PMI or Scrum, through an adaptation rule.

The input of project knowledge and the maintenance of the case base could be automated through interfaces to operational (software) applications (e.g., Sharepoint and SAP systems). The interfaces to the business applications could be implemented as independent serverless functions in a cloud environment that retrieve data from the business applications and display it in an ontology-supported CBR system for a cloud-native application. A further investigation could examine the extent to which it is possible to automate the input of project knowledge, or at least to automate the pre-processing of project knowledge as a first step. Case entry could also be accelerated by generating templates. AI techniques could be used to recognize patterns for case templates. It would be conceivable to use Doc2Vec, described later in the outlook, to recognize patterns in a large number of documents and use them to create case templates.

The CBR tool jCORA represents a different starting point for further developments aimed at overcoming the current limitations and preparing the CBR tool for “future-proof” information technology. In this article, we presented the reading and processing of an ontology, the similarity algorithm with specific similarity functions, and a user interface as a click prototype designed for use as a cloud-native application as examples. In a more in-depth investigation, it would be desirable to provide all the areas considered in isolation as a fully functional cloud-native application and to extend them with additional specific similarity functions.

As part of further development as a cloud-native application, it is also conceivable to use the Python module OWLReady2 to extend ontology-supported case-based reasoning with ontology editing functions, for example to avoid always having to resort to the ontology editor Protégé when making necessary additions to an ontology.

Another starting point for further investigations is the Word2Vec algorithm, which was used for the two specific similarity functions `simStringBOS` and `simPreTrained`. Various further developments are possible here. On the one hand, the Word2Vec algorithm can be improved with other Python modules (e.g., Tensorflow) in terms of training so that the artificial neural network can be expanded to include network levels. Furthermore, it would be desirable to use a transformer algorithm in addition to a Word2Vec algorithm and to make this available as a specific similarity function. Although there currently exist no modules that provide a similar result to ChatGPT or Bidirectional Encoder Representations from Transformers (BERT), Tensorflow can be used to create a basic application with the Transformer approach in a Python project. Documentation on how a general Transformer approach can be used in a Python project can be found in TENSORFLOW (2023). This documentation describes the use of the Transformer approach for the translation of natural language texts as an example. A content-related similarity analysis, e.g., between sentences or words, is also possible, but would need to be evaluated in a further study. Current publications show these possibilities; cf. for example BAER/PURVES (2023), p. 56; WORTH (2023), p. 14; ZHANG et al. (2023), p. 3. The sources mentioned describe the basic suitability of the transformer approach for checking the similarity of natural language texts.

In an in-depth study, it would be desirable to implement these approaches as independent specific similarity functions as serverless functions and to integrate them into an ontology-supported CBR system for a cloud-native application. The following research questions could be formulated for follow-up studies:

- How can the Transformer approach be integrated into an ontology-based CBR system?
- How can ontologies be used in combination with transformer algorithms to develop specific adaptation rules?

In this article, we used only two specific similarity functions based on the Word2Vec technique (`simStringBOS` und `simPreTrained`). In further investigations, additional specific similarity functions could be implemented that are trained with different documents. Based on the Word2Vec calculation model, the Doc2Vec submodule exists in Gensim, which offers a promising approach for comparing documents. Doc2Vec makes

it possible to determine the similarity of content between documents. The following research questions can be answered with Doc2Vec:

- Which two documents from a set of documents are most similar to each other?
- Can paragraphs, passages, or individual sentences be found that are similar to other paragraphs, passages, or individual sentences in the same document or other documents?

Furthermore, a combination of Word2Vec, Transformer approach, and Doc2Vec for ontology-supported case-based reasoning could represent an interesting research approach. It could be investigated whether additional evaluation options exist through combined use, for example to make predictions of project decisions. With regard to the prediction of project decisions, the approach differs from the reuse of experience-based knowledge in that individual project decisions can possibly be predicted on the basis of past project experience. In this case, the focus is on individual project decisions and not on the entire project. The possibility of prediction using the Word2Vec algorithm is already the subject of initial investigations. For example, the Word2Vec algorithm is being used to develop models for predicting the severity of software errors; cf. AGRAWAL/GOYAL (2021), pp. 106–108. Regarding the improvement of purchasing forecasts using the Word2Vec algorithm, see ESMELI/BADER-EL-DEN/ABDULLAHI (2020), pp. 2–5. For the detection of anomalies in system logs using the Word2Vec algorithm, see WANG et al. (2022), pp. 1210–1220.

Word2Vec could also offer possibilities for the definition of adaptation rules, the automated generation of ontologies, and the verification of an ontology. Initial research approaches already exist in this area. For the possible automated generation of an ontology using Word2Vec, please refer to YOUN/NARAVANE/TAGKOPOULOS (2020), pp. 2–3; MAHMOUD/ELBEH/ABDLKADER (2018), pp. 184–188; WOHLGENANT/MINIC (2016), pp. 2–4. We present YOUN/NARAVANE/TAGKOPOULOS as an example, as the authors use a similar approach to this article. These authors use a text corpus (including Wikipedia entries and food databases) to create an ontology for food; cf. YOUN/NARAVANE/TAGKOPOULOS (2020), pp. 2–3. In addition, reference is made to CHEN et al. (2021), pp. 1815–1843, who develop the innovative OWL2Vec approach based on Word2Vec to generate an automated ontology embedding.

Word2Vec could be a tool for the automated pre-processing of project knowledge from natural language texts in order to automatically extract attribute and relation values of cases for the representation of safety-critical IT projects. BEIBEL (2011), p. 219, has already addressed this aspect as a possible research approach. However, he saw limitations with regard to the available text analysis tools. It would be desirable to check whether

the limitations mentioned by BEIßEL can be overcome using a Word2Vec or Transformer approach.

The adaptation of cases in the context of ontology-supported case-based reasoning represents a particularly serious problem area. In this article, we have described a conceptual approach for the acquisition of adaptation knowledge. In a further study, however, it would be desirable to consider the adaptation of cases in the context of ontology-supported case-based reasoning. In particular, the implementation of adaptation rules, the automated acquisition of said rules, and their application represent an interesting field of research in which the techniques used here, such as SWRL rules, Word2Vec, and the transformer approach, could provide possible support. In the context of public procurement procedures (from the perspective of a potential contractor), for example, such support could consist of the automated processing of service descriptions on relevant tendering platforms using the Transformer or Word2Vec approach and comparing them with existing experience-based knowledge. This enables the identification of tenders with a higher probability of winning, as experience is already available, and solution approaches for the tendered service are available or can be adapted to the new problem (the tendered service) using adaptation rules. In this way, a company's sales process can be strengthened by using automation to ensure that only those tenders are processed where experience is available, or even by applying customization rules using SWRL, Word2Vec, or transformer systems to solve the new problem.

Bibliography

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, pp. 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. (eds.): *On the move to meaningful internet systems 2006: OTM 2006 workshops – OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, COMINF, IS, KSinBIT, MI-OSCIAO, 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, pp. 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. (eds.): *KI 2009 – Advances in Artificial Intelligence: 32nd Annual German Conference on AI*, Paderborn, Germany, September 15-18, 2009, *Proceedings*. Berlin – Heidelberg (2009), pp. 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), pp. 104–115.

AICHELE (2021)

Aichele, C.: Künstliche Intelligenz für klein- und mittelständische Unternehmen. In: Aichele, C.; Herrmann, J. (eds.): *Betriebswirtschaftliche KI-Anwendungen – Digitale Geschäftsmodelle auf Basis Künstlicher Intelligenz*. Wiesbaden 2021, pp. 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), No. 1, pp. 79–97.

AMAZON (2022)

Amazon: Annual Report 2021. Published on 03.02.2022. https://s2.q4cdn.com/299287126/files/doc_financials/2022/ar/Amazon-2021-Annual-Report.pdf, last accessed on 11.04.2023.

AMAZON WEB SERVICES, INC. (2022a)

Amazon Web Services, Inc.: Was ist Python? – Python-Leitfaden für Cloud-Einsteiger – AWS. Copyright 2022. <https://aws.amazon.com/de/what-is/python/>, last accessed on 11.12.2022.

AMAZON WEB SERVICES, INC. (2022b)

Amazon Web Services, Inc.: AWS Cloud9 Amazon Web Services. Copyright 2022. <https://aws.amazon.com/de/cloud9/>, last accessed on 11.12.2022.

AMAZON WEB SERVICES, INC. (2022c)

Amazon Web Services, Inc.: Customers. Copyright 2022. 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, last accessed on 11.12.2022.

AMAZON WEB SERVICES, INC. (2022d)

Amazon Web Services, Inc.: AWS Documentation. Copyright 2022. https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do, last accessed on 11.12.2022.

AMAZON WEB SERVICES, INC. (2022e)

Amazon Web Services, Inc.: Globale AWS Infrastruktur & Availability Zones – AWS. Copyright 2022. <https://aws.amazon.com/de/about-aws/global-infrastructure/>, last accessed on 11.12.2022.

ANGERMEIER (2016)

Angermeier, G.: PDCA-Zyklus. <https://www.projektmagazin.de/glossarterm/pdca-zyklus>, last accessed on 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. (eds.): Successful Case-based Reasoning Applications – I, Berlin – Heidelberg 2010, pp. 97–116.

AVESANI/SUSI (2010)

Avesani, P.; Susi, A.: Case-Based Ranking for Environmental Risk Assessment. In: Kacprzyk, J.; Montani, S.; Jain, L. C. (eds.): Successful Case-based Reasoning Applications – I. Berlin – Heidelberg 2010, pp. 165–185.

AXELOS (2015)

Axelos: PRINCE2 Agile. Norwich 2015.

AXELOS (2018)

Axelos: Erfolgreiche Projekte managen mit PRINCE2. 6th ed., 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), No. 1, pp. 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, contribution-specific pagination: pp. 1–34.

BEIBEL (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. (eds.): Organisatorische Innovationen mit Good Governance und Semantic Knowledge Management in Logistik-Netzwerken – Wissenschaftliche Grundlagen und Praxisanwendungen. Berlin 2015, pp. 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, pp. 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), No. 1, pp. 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. Published in 2021. https://www.capgemini.com/de-de/wp-content/uploads/sites/5/2021/02/Studie_IT-Trends_2021_Capgemini.pdf, last accessed on 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. Published in 2022. <https://spectrum.ieee.org/top-programming-languages-2022>, last accessed on 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), pp. 1813–1845.

DEBELLIS (2021)

DeBellis, M.: A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins. 2021.

DELOITTE (2022)

Deloitte: TechTrends 2022. Published in 2022. https://www2.deloitte.com/content/dam/Deloitte/pt/Documents/tech-trends/tech-trends-2022/DI_Tech-trends-2022.pdf, last accessed on 11.04.2023.

DER BUNDESRAT DER SCHWEIZER REGIERUNG (2021)

Der Bundesrat der Schweizer Regierung: Herausforderungen bei zwei wichtigen Projekten zur sicheren Kommunikation. Published on 16.02.2021. <https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-82340.html>, last accessed on 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. (eds.): 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, pp. 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), No. 11, pp. 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. (ed.): Complex, Intelligent and Software Intensive Systems – Proceedings of the 16th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2022). Cham 2022, pp. 425–434.

DIVARI (2011)

Divari, V.: Konzeption und Entwicklung eines Verfahrens zum Matching von Prozessmodellen. Published on 18.07.2011. https://elib.uni-stuttgart.de/bitstream/11682/2754/1/DIP_3149.pdf, last accessed on 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, pp. 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. (ed.): Intelligent Systems and Applications: Proceedings of the 2022 Intelligent Systems Conference (IntelliSys) Volume 1. Cham 2023, pp. 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. (eds.): Model-Driven Engineering and Software Development – 4th International Conference, MODELSWARD 2016, Rome, Italy, February 19-21, 2016, Revised Selected Papers. Cham 2017, pp. 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, contribution-specific pagination: pp. 1–8.

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 (2020)

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, 2023-2030. Published in 2020. <https://www.fortunebusinessinsights.com/cloud-computing-market-102697>, last accessed on 11.06.2023.

FOWLER/LEWIS (2015)

Fowler, M.; Lewis, J.: Microservices – Nur ein weiteres Konzept in der Softwarearchitektur oder mehr? In: Objektspektrum, No. 1/2015, pp. 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. (eds.): 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, pp. 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. https://www.tech-nova-cpi.org/images/Documenti-pdf/Top%20Strategic%20Technology%20Trends%20for%202022_Gartner_31gen2022.pdf, last accessed on 11.04.2023.

GARTNER (2022a)

Gartner: Magic Quadrant for Cloud AI Developer Services. Published on 23.05.2022. <https://www.gartner.com/en/documents/4014812>, last accessed on 11.04.2023.

GARTNER (2022b)

Gartner: Magic Quadrant for Cloud Infrastructure and Platform Services. Copyright 2022. <https://www.gartner.com/technology/media-products/reprints/AWS/1-271W1O-SP-DEU.html>, last accessed on 09.12.2022.

GASSMANN (2001)

Gassmann, O.: High-Risk-Projekte als Erfolgsfaktor in dynamischen Industrien. In: Gassmann, O.; Kobe, C.; Voit, E. (eds.): High-Risk-Projekte. Berlin – Heidelberg 2001, pp. 3–23.

GITHUB (2022a)

GitHub: GitHub - RaRe-Technologies/genism – Topic Modelling for Humans. Published in 2022. <https://github.com/RaRe-Technologies/gensim>, last accessed on 17.12.2022.

GITHUB (2022b)

GitHub: GitHub – RaRe-Technologies/gensim-data: Data repository for pretrained NLP models and NLP corpora. Published in 2022. <https://github.com/RaRe-Technologies/gensim-data>, last accessed on 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. <https://arxiv.org/abs/1402.3722>, last accessed on 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. <https://colab.research.google.com/>, last accessed on 11.04.2023.

GOOGLE CLOUD (2020)

Google Cloud: Das neue Grace-Hopper-Seekabel zwischen den USA, dem Vereinigten Königreich und Spanien. Published on 21.07.2020. <https://cloud.google.com/blog/de/products/infrastruktur/das-neue-grace-hopper-seekabel-von-google>, last accessed on 29.07.2023.

GOOGLE CLOUD (2022a)

Google Cloud: Produkte und Dienste der Google Cloud. Copyright 2022. <https://cloud.google.com/products>, last accessed on 19.12.2022.

GOOGLE CLOUD (2022b)

Google Cloud: Weltweite Standorte – Regionen und Zonen der Google Cloud. Copyright 2022. <https://cloud.google.com/about/locations?hl=de#regions>, last accessed on 12.19.2022.

GOOGLE DEVELOPERS (2022)

Google Developers: Python. Copyright 2022. <https://developers.google.com/learn/topics/python>, last accessed on 11.12.2022.

GRUBER (1993)

Gruber, T. R.: A translation approach to portable ontology specifications. In: Knowledge Acquisition, Vol. 5 (1993), pp. 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), No. 6, pp. 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. (eds.): Open Semantic Technologies for Intelligent Systems – 11th International Conference, OSTIS 2021, Minsk, Belarus, September 16–18, 2021, Revised Selected Papers. Cham 2022, pp. 361–372.

HABERMANN (2013)

Habermann, F.: Hybrides Projektmanagement – agile und klassische Vorgehensmodelle im Zusammenspiel. In: HMD Praxis der Wirtschaftsinformatik, Vol. 50 (2013), pp. 93–102.

HARRIS (2022)

Harris, C.: Microservices und monolithische Architektur im Vergleich. Published in 2022. <https://www.atlassian.com/de/microservices/microservices-architecture/microservices-vs-monolith>, last accessed on 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), No. 5, pp. 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. (eds.): Projektmanagement und Vorgehensmodelle 2014 – Soziale Aspekte und Standardisierung. Bonn 2014, pp. 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, pp. 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, Article 30, contribution-specific pagination: pp. 1–19.

INITIATIVE D21 (2022)

Initiative D21: D21-Digital-Index 2021/2022: Jährliches Lagebild zur Digitalen Gesellschaft. Vertiefungsthema – Digitale Nachhaltigkeit. https://initiatived21.de/app/uploads/2022/02/d21-digital-index-2021_2022.pdf, last accessed on 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. <https://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-1%20Standard.pdf>, last accessed on 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. (eds.): Manipulationssichere Cloud-Infrastrukturen – Nachhaltige Digitalisierung durch Sealed Cloud Security. Wiesbaden 2020, pp. 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), No. 3, pp. 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, pp. 43–52.

KALS (2021)

Kals, J.: Qualitätsmanagement (QM). In: Wannenwetsch, H. (ed.): Integrierte Materialwirtschaft, Logistik, Beschaffung und Produktion – Supply Chain im Zeitalter der Digitalisierung. Berlin 2021, pp. 273–321.

KANTARWORLD PANEL (2022)

Kantarworldpanel: Android vs. iOS – Smartphone OS sales market share evolution. Copyright 2022. <https://www.kantarworldpanel.com/global/smartphone-os-market-share/>, last accessed on 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, pp. 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, pp. 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. (eds.): Neue Algorithmen für praktische Probleme – Variationen zu Künstlicher Intelligenz und Künstlichem Leben. Wiesbaden 2021, pp. 9–20.

KLÜVER/KLÜVER/SCHMIDT (2021)

Klüver, C.; Klüver, J.; Schmidt, J.: Modellierung komplexer Prozesse durch naturanaloge Verfahren – Künstliche Intelligenz und Künstliches Leben. 3rd ed., 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! Published on 24.08.2023. <https://www.wiwo.de/technologie/digitale-welt/daten-diebstaehe-bei-cloudanbietern-sicherheit-in-der-cloud-ist-nur-eine-illusion/29352358.html>, last accessed on 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. (eds.): FinTech – Digitalisierung, Künstliche Intelligenz und aufsichtsrechtliche Regulierung von Finanzdienstleistungen. 2nd ed., Berlin 2022, pp. 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), pp. 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. Published on 04.04.2023. <https://readthedocs.org/projects/owlready2/downloads/pdf/latest/>, last accessed on 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. (eds.): Data-Driven Process Discovery and Analysis. Berlin – Heidelberg 2012, pp. 189–211.

LINTHICUM (2022)

Linthicum, D.: 3 Gründe – Warum Sie mit der Cloud kein Geld sparen. Published on 30.09.2022. <https://www.cio.de/a/warum-sie-mit-der-cloud-kein-geld-sparen,3612872>, last accessed on 11.04.2023.

LÜNENDONK (2021)

Lünendonk: Cloud-native Software Development – Mit Cloud-Technologien und Agilität zu mehr Innovationsgeschwindigkeit und Wettbewerbsvorteilen. Lünendonk-Studie. Published in 2021. <https://www.luenendonk.de/produkte/studien-publikationen/luenendonk-studie-2021-cloud-native-software-development-it/>, last accessed on 11.04.2023.

LUONG/PHAM/MANNING (2015)

Luong, M.-T.; Pham, H.; Manning, C. D.: Effective Approaches to Attention-based Neural Machine Translation. Published on 20.09.2015. <https://arxiv.org/abs/1508.04025>, last accessed on 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, pp. 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), No. 4, pp. 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? Status: 2018, Copyright 2024. <https://www.mend.io/most-secure-programming-languages/>, last accessed on 10.11.2024.

MICROSOFT (2023a)

Microsoft: Was sind Azure-Regionen und -Verfügbarkeitszonen? Published on 16.03.2023. <https://learn.microsoft.com/de-de/azure/availability-zones/az-overview>, last accessed on 11.04.2023.

MICROSOFT (2023b)

Microsoft: Azure-Produkte. Azure Produkte suchen oder durchsuchen. Copyright 2023. <https://azure.microsoft.com/de-de/products/>, last accessed on 11.04.2023.

MIKOLOV et al. (2013a)

Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.: Efficient Estimation of Word Representations in Vector Space. Published on 07.09.2013. <https://arxiv.org/abs/1301.3781>, last accessed on 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. Published on 16.10.2013. <https://arxiv.org/abs/1310.4546>, last accessed on 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. (eds.): 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, pp. 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, pp. 2392–2399.

NIGHOJKAR/KHLYZOVA/LICATO (2022)

Nighojkar, A.; Khlyzova, A.; Licato, J.: Cognitive Modeling of Semantic Fluency Using Transformers. Published on 20.08.2022. <https://arxiv.org/abs/2208.09719>, last accessed on 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. (eds.): Case-Based Reasoning Research and Development. Cham 2020, pp. 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. (eds.): Case-Based Reasoning Research and Development. Cham 2022, pp. 125–139.

NOY/MCGUINNESS (2001)

Noy, F.; McGuinness, D.: Ontology Development 101 – A Guide to Creating Your First Ontology. Published in 2001. https://protege.stanford.edu/publications/ontology_development/ontology101.pdf, last accessed on 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. (eds.): Emerging Technologies in Data Mining and Information Security – Proceedings of IEMIS 2022, Volume 1. Singapore 2023, pp. 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, pp. 979–1005.

OLIVEIRA ROCHA (2021)

Oliveira Rocha, H. F.: Practical Event-Driven Microservices Architecture – Building Sustainable and Highly Scalable Event-Driven Microservices. Berkeley 2021.

PAAB/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. (eds.): Data Management, Analytics and Innovation – Proceedings of ICDMAI 2022. Singapore 2023, pp. 291–302.

PERTLWIESER (2022)

Pertlwieser, M.: Das richtige Digitalisieren – Eine ‚Masterclass‘ zum digitalen Wandel für Manager:innen und Unternehmer:innen. Wiesbaden 2022, pp. 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. <https://pypl.github.io/PYPL.html>, last accessed on 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. Published in 2002. <https://www.python.org/psf/mission/>, last accessed on 11.04.2023.

PYTHON (2023a)

Python: Python Software Foundation. Copyright 2023. <https://www.python.org/psf/>, last accessed on 11.04.2023.

PYTHON (2023b)

Python: What is Python? Executive Summary. Copyright 2023. <https://www.pyhon.org/doc/essays/blurb/>, last accessed on 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, pp. 99–113.

RASCHKA/MIRJALILI (2019)

Raschka, S.; Mirjalili, V.: Python Machine Learning – Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2. 3rd ed., 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. (eds.): Applications and Innovations in Intelligent Systems XIV. London 2007, pp. 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), pp. 126–145.

REGENFUß/NINK (2022)

Regenfuß, T./Nink, T.: Was kann die Google Cloud? Published on 05.10.2022. <https://www.cio.de/a/was-kann-die-google-cloud,3667814#:~:text=In%20der%20Google%20Cloud%20erfolgt,kostenoptimierten%20Zuschnitt%20der%20virtuellen%20Server>, last accessed on 11.04.2023.

ŘEHŮŘEK (2011)

Řehůřek, R.: Scalability of semantic analysis in natural language processing. Dissertation Masaryk University. Brunn 2011.

ŘEHŮŘEK (2022)

Řehůřek, R.: API Reference. Word2vec embeddings. Published on 21.12.2022. <https://radimrehurek.com/gensim/models/word2vec.html>, last accessed on 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, pp. 46–50.

RICHTER (2008)

Richter, M.: Similarity. In: Perner, P. (ed.): Case-Based Reasoning on Images and Signals. Berlin – Heidelberg (2008), pp. 25–90.

RONG (2014)

Rong, X.: word2vec Parameter Learning Explained. Published on 11.11.2014. <https://arxiv.org/abs/1411.2738>, last accessed on 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, pp. 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, contribution-specific pagination: pp. 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. Published on November 2020. <https://scrumguides.org/docs/scrum-guide/v2020/2020-Scrum-Guide-US.pdf>, last accessed on 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. Published on 19.08.2016. <https://www.parlament-berlin.de/ados/17/Haupt/vorgangh17-2941-v.pdf>, last accessed on 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 Wirtschaftswissenschaften, 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, contribution-specific pagination: pp. 1–7.

SLASHDATA (2021)

SlashData: State-of-Cloud-Native-Development 2021 – The latest trends from our Q3 2021 survey of 19000 developers. Published in 2021. https://www.cncf.io/wp-content/uploads/2022/05/Q3-2021-State-of-Cloud-Native-development_FINAL.pdf, last accessed on 11.04.2023.

SLASHDATA (2022)

SlashData: State of the Developer Nation – 22nd Edition – The latest trends from our Q1 2022 survey of 20000 developers. Published in 2022. https://slashdata-website-cms.s3.amazonaws.com/sample_reports/VZtJWxZw5Q9NDSAQ.pdf, last accessed on 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, pp. 74–78.

STAAB (2011)

Staab, S.: Ontologies and Similarity. In: Ram, A.; Wiratunga, N. (eds.): Case-based reasoning research and development – 19th International Conference on Case-Based Reasoning, ICCBR 2011, London, UK, September 12-15, 2011, Proceedings. Berlin 2011, pp. 11–16.

STACK OVERFLOW (2021)

Stack Overflow: Stack Overflow Developer Survey 2021. Published in 2021. <https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language>, last accessed on 11.04.2023.

STACK OVERFLOW (2023)

Stack Overflow: Newest owlready Questions. Copyright 2023. <https://stackoverflow.com/questions/tagged/owlready>, last accessed on 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, pp. 471–482.

SYNERGY RESEARCH GROUP (2022)

Synergy Research Group: Q2 Cloud Market Grows by 29% Despite Strong Currency Headwinds; Amazon Increases its Share. Published in 2022. <https://www.srgresearch.com/articles/q2-cloud-market-grows-by-29-despite-strong-currency-headwinds-amazon-increases-its-share>, last accessed on 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, contribution-specific pagination: pp. 1–12.

TAYLOR/DU PREEZ (2023)

Taylor, R. M. C.; Du Preez, J. A.: SimLDA – A Tool for Topic Model Evaluation. In: Arai, K. (eds.): Proceedings of the Future Technologies Conference (FTC) 2022. Cham 2023, pp. 534–554.

TENSORFLOW (2023)

TensorFlow: Neural machine translation with a Transformer and Keras. Copyright 2023. <https://www.tensorflow.org/text/tutorials/transformer>, last accessed on 11.04.2023.

TENZER (2022)

Tenzer, F.: Anteil der privaten Haushalte in Deutschland mit einem Mobiltelefon von 2000 bis 2022. Published on 03.11.2022. <https://de.statista.com/statistik/daten/studie/198642/umfrage/anteil-der-haushalte-in-deutschland-mit-einem-mobiltelefon-seit-2000/>, last accessed on 11.04.2023.

URBAN/GARLOFF (2022)

Urban, M.; Garloff, K.: Sovereign Cloud Stack. In: Datenschutz und Datensicherheit, Vol. 46 (2022), No. 10, pp. 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. Published on 12.06.2017. <https://arxiv.org/abs/1706.03762>, last accessed on 05.05.2023.

VETTOR/SMITH (2023)

Vettor, R.; Smith, S.: Architecting Cloud-Native .NET Apps for Azure. Copyright 2023. <https://learn.microsoft.com/pdf?url=https%3A%2F%2Flearn.microsoft.com%2Fen-us%2Fdotnet%2Farchitecture%2Fcloud-native%2Ftoc.json>, last accessed on 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, pp. 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), pp. 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. (eds.): 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, pp. 497–506.

WOHLGENANNT/MINIC (2016)

Wohlgenannt, G.; Minic, F.: Using word2vec to Build a Simple Ontology Learning System. Published in 2016. <http://ceur-ws.org/Vol-1690/paper37.pdf>, last accessed on 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), pp. 1–21.

WYNER (2008)

Wyner, A.: An ontology in OWL for legal case-based reasoning. In: Artificial Intelligence and Law, Vol. 16 (2008), pp. 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, pp. 584–784 (contribution-specific pagination in this article: pp. 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. (eds.): *Ontologiebasierte Kompetenzmanagementsysteme – Grundlagen, Konzepte, Anwendungen*. Berlin 2005, pp. 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. (eds.): *Digitalisierung und Nachhaltigkeit – Transformation von Geschäftsmodellen und Unternehmenspraxis*. Berlin – Heidelberg 2022, pp. 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. (eds.): *Corporate Semantic Web*. Berlin – Heidelberg 2015, pp. 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. (eds.): *Organisatorische Innovationen mit Good Governance und Semantic Knowledge Management in Logistik-Netzwerken*. Berlin 2015, pp. 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), No. 1, pp. 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), pp. 57–64.



of public administration, with a special focus on the protection and implementation of critical infrastructure.

Mr. Ganen Sethupathy, PhD., holds a Master's degree (M.Sc.) in Geoinformatics and currently heads a department within an IT consulting group. Alongside his professional role, he completed an MBA, which took him to cities such as New York and San Diego. He later earned his Ph.D. at the University of Duisburg-Essen as part of the "KI-LiveS" research and transfer project, funded by the German Federal Ministry of Education and Research (BMBF), focusing on practical applications of the AI technology "ontology-supported case-based reasoning." His research interests include ontologies, case-based reasoning, and artificial neural networks. Dr. Sethupathy is also furthering his expertise in Artificial Intelligence through a three-year program on Machine Learning and AI at the Massachusetts Institute of Technology (MIT). He is particularly committed to advancing the digitalization



Mr. Jan Peter Schagen, M.Sc., was a research assistant at the Institute for Production and Industrial Information Management and is doctoral student at the Faculty of Economics of the University of Duisburg-Essen. He studied Business Administration (B.Sc.) at the University of Duisburg-Essen, Campus Essen, followed by a Master's degree in Markets and Companies (M.Sc.). As part of his work as a research assistant, he was involved in the joint project KI-LiveS, which aimed to transfer techniques from artificial intelligence research into business practice. His dissertation project focuses on the "intelligent" reuse of empirical knowledge in project management using ontology-supported case-based reasoning with a particular focus on the design, implementation and evaluation of adaptation rules for an ontology-supported case-based reasoning system.



administration and business informatics with a focus on project management, logistics, supply chain management and production planning and control. On the other hand, he is intensively involved in the transfer of findings from research into artificial intelligence to economic problems, particularly with regard to knowledge-based systems, ontologies and case-based reasoning.

Mr. Stephan Zelewski, PhD., was Professor of Business Administration and Director of the Institute for Production and Industrial Information Management at the Faculty of Economics at the University of Duisburg-Essen (until November 2024). He studied business administration and economics (Dipl.-Kfm., Dipl.-Volksw.) at the Universities of Münster and Cologne. In 1985, he received his doctorate from the University of Cologne with a thesis on the business application potential of artificial intelligence. After his habilitation on structuralist production theory, he was Professor of Business Administration and Director of the Institute for Production Management and Industrial Information Management at the University of Leipzig from 1993 to 1998. Mr. Zelewski's main areas of work are, on the one hand, issues of computer-aided production management at the interface between business

In safety-critical IT projects, the success of which is crucial to public safety, the failure of an IT system can have far-reaching consequences. In practice, however, the reuse of valuable experience-based knowledge, which is often available in natural language and on different IT systems or in physical documents, poses a considerable challenge. If this experience-based knowledge is not taken into account, the consequences can be serious and lead to high additional costs and considerable project delays.

One possible approach to solving these problems lies at the interface of business administration, business informatics and computer science. This approach combines the previously separate disciplines of project management, knowledge management and artificial intelligence (AI). It shows how proven “traditional” AI techniques, especially ontologies and case-based reasoning, can be used effectively in safety-critical IT projects. By combining ontology-supported case-based reasoning with artificial neural networks for the “modern” Word2Vec technique, a cloud-based software is designed that can be used to significantly improve the reuse of experience-based knowledge in safety-critical IT projects.

The article presented here shows how the combination of traditional and modern AI techniques can provide a key to the systematic reuse of mission-critical experience-based knowledge in IT projects.