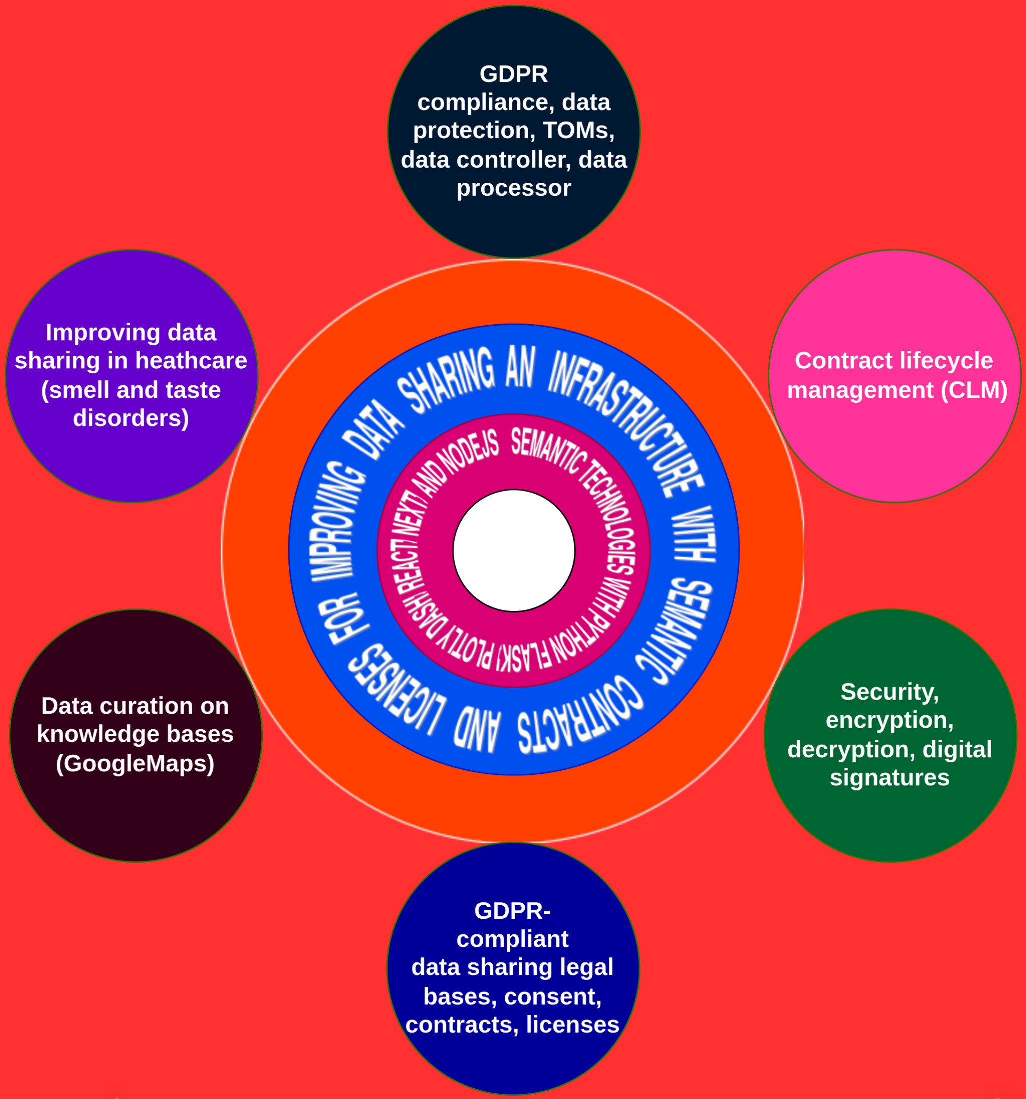
AN INFRASTRUCTURE WITH SEMANTIC CONTRACTS AND LICENSES FOR IMPROVING DATA SHARING


GDPR compliance, data protection, TOMs, data controller, data processor
Improving data sharing in heathcare (smell and taste disorders)
Contract lifecycle management (CLM)
IMPROVING DATA SHARING AN INFRASTRUCTURE WITH SEMANTIC CONTRACTS AND LICENSES FOR
SEMANTIC TECHNOLOGIES WITH PYTHON FLASK, PLOTLY DASH, REACT/ NEXTJ AND NODEJS
Data curation on knowledge bases (GoogleMaps)
Security, encryption, decryption, digital signatures
GDPR-compliant data sharing legal bases, consent, contracts, licenses

Amar Tauqeer

Propositions

1. Ontology is a good option as a data model for constructing and modeling data sharing contracts.
(this thesis)
2. Automated General Data Protection Regulation (GDPR) compliance verification checks are necessary to identify contract violations or breaches.
(this thesis)
3. Decision-making improves by eliminating repetitive tasks for medical specialists in the modern age.
4. Applying FAIR (Findable, Accessible, Interoperable, and Reusable) data principles on datasets containing personal or identifying (name, address) information require additional data privacy regulations.
5. Automated and semantic contracting would have many implications on the way we work and live.
6. The awareness of data privacy has a significant impact on social life.

Propositions belonging to the thesis, entitled

An Infrastructure with Semantic Contracts and Licenses
for Improving Data Sharing

Amar Tauqeer
Wageningen, 02 July 2024

An Infrastructure with Semantic Contracts and Licenses for Improving Data Sharing

Amar Tauqeer

Thesis committee

Promotor
Prof. Dr Anna Fensel
Personal Chair at the Artificial Intelligence Group
Wageningen University & Research

Other members
Prof. Dr R. da Silva Torres, Wageningen University & Research
Dr S.M. Ní Fhlaithearta, Wageningen University & Research
Dr J.J. Koehorst, Wageningen University & Research
Dr B. Öztürk, Wageningen University & Research

This thesis was conducted under the auspices of the Wageningen School of Social Sciences (WASS)

An Infrastructure with Semantic Contracts and Licenses for Improving Data Sharing

Amar Tauqeer

Thesis
submitted in fulfillment of the requirements for the degree of doctor at
Wageningen University
by the authority of the Rector Magnificus
Prof. Dr C. Kroeze,
in the presence of the
Thesis Committee appointed by the Academic Board
to be defended in public
on 2 July 2024
at 11 a.m. in the Omnia Auditorium

Amar Tauqeer
An Infrastructure with Semantic Contracts and Licenses for Improving Data Sharing
251 pages.

PhD thesis, Wageningen University, Wageningen, NL (2024)
With references, with summary in English

ISBN 978-94-6510-003-6
DOI: *https://doi.org/10.18174/653055*

Abstract

Data sharing and reusing for business and research now become typical practices. Data sharing demand is increasing frequently due to most industries shifting their businesses to digital form, particularly in domains, such as healthcare, smart cities, and insurance. Data sharing, the generation of a huge volume of data, and data across a variety of sources are some key factors crucial to the modern digital economy. With these key factors, there have been several privacy and data misuse concerns. For example, the General Data Protection Regulation (GDPR) is the most significant data privacy regulation, effected on May 25, 2018. For lawful data processing, GDPR defines six legal bases and consent is the most used legal base. However, in scenarios where consent is not enough, selling of data, for example, a contract is required, where additional obligations are defined to get more control over data. Contracts have many implementation aspects, specifically, the GDPR and the complexity of contract lifecycle management (CLM), which are most important nowadays. GDPR rights must be bound legally with contracts for lawful data processing. Over the last decade, research in creating contracts has gained popularity but complying with GDPR that could be used as a standard has not been elaborated sufficiently yet. The CLM is a complex, time, and effort-consuming process consisting of stages such as drafting, negotiating, and signing that require more security. For improving and providing the highest security to CLM, there is still a need to explore the signing stage with digital signatures. Besides contracts, digital assets' licensing is another key factor in improving data sharing in the digital economy. The clearance of rights manually is a time-consuming and error-prone process. The Data License Clearance Center (DALICC) and License Clearance Tool (LCT) are a few solutions for the clearance of rights on digital assets that save not only costs but also time. There is still a need to explore the integration and execution of the aforementioned solutions with contracts. Licenses are not part of GDPR but have legal regulations. A GDPR-compliant contract with a standard license can become another legal base for lawful data processing. This helps contractors to gain more control over data by combining licenses and additional contractual obligations. The main goal of this research is to improve data sharing infrastructure with contracts and licenses using semantic technologies. The proposed solution improves the CLM creation process, secures the signing process with digital signatures, supports the automated detection of contract breaches, and enriches the semantic models of contracts and licenses that can be interpretable by humans and machines. Further, the solution is tested and evaluated against real-world use cases (UC) in the smashHit [1]

[1]https://cordis.europa.eu/project/id/871477

project in the domains of smart cities and insurance, which demonstrates the feasibility of the approach. Further, two other UCs in the domains of healthcare and tourism were also investigated, which are shown as enablers for data sharing to ensure better user interaction with the data sharing systems and data quality (validation).

Contents

List of Acronyms

B2B Business to Business.

B2C Business to Consumer.

CCV Contract Compliance Verification.

CDMM Consent and Data Management Mode.

CLM Contract Lifecycle Management.

CSTD Clinical Smell and Taste Disorders Ontology.

DALICC Data Licenses Clearance Center.

DBMS Database Management System.

DC Data Controller.

DCAT Data Catalog Vocabulary.

DP Data Processor.

DPV Data Privacy Vocabulary.

DPO Data Protection Officer.

DS Data Subject.

DSQA Domain Specific QA System for Smart Health Based on a Knowledge Graph.

EHR Electronic Health Records.

FactBench Fact Validation Benchmark.

FIBO Financial Industry Business Ontology.

GDPR General Data Protection Regulation.

KGs Knowledge Graph.

KV Knowledge Validation.

LCT License Clearance Tool.

LloPY Legal Compliant Ontology to Preserve Privacy for the Internet of Things.

MedDRA Medical Dictionary for Regulatory Activities Terminology.

MTCO Multi-Tier Contract Ontology.

NIST National Institute of Standards and Technology.

OEMs Original Equipment Manufacturers.

OOPS! OntOlogy Pitfall Scanner.

OWL Web Ontology Language.

QA Question Answering.

QAKG Question Answering over KG.

RDF Resource Description Framework.

RDFS RDF Schema.

RQ Research Question.

SMEs Small and Medium Enterprises.

SPARQL Simple Protocol and Resource Description Framework Query Language.

SQL Structured Query Language.

SSN Semantic Sensor Network ontology.

SRQ Sub Research Question.

SW Semantic Web.

TAs Trusted Authorities.

TKG Tirol Knowledge Graph.

TOMs Technical and Organisational Measures.

UC	Use Case.
UI	User Interface.
URI	Uniform Resource Identifier.
W3C	World Wide Web Consortium.
ZKP	Zero-Knowledge Proof.

List of Listings

List of Figures

List of Tables

CHAPTER 1

General Introduction

1.1 Introduction

Information sharing and reusing for business and research now become typical practices. These serve as the foundation for the digital economy (Stott, 2014). Mostly, industries have shifted their businesses to digital form, particularly in the domain of healthcare, e-commerce, education, smart cities, insurance, and sports (Cai et al., 2023; Cao et al., 2015; Jussen et al., 2023; Susha et al., 2023). By this, data sharing demand is increasing frequently. One of the examples where data sharing played a crucial role is trusted Internet of Things (IoT) data sharing through smart contracts in the domain of agriculture (Taleka et al., 2023). A smart contract is based on blockchain technology and defined as “A smart contract is a computer program or a transaction protocol that is intended to automatically execute, control or document events and actions according to the terms of a contract or an agreement” (Fries, Martin AND Paal, Boris P., 2019; Röscheisen et al., 1998; Savelyev, 2017). This minimized trust factors like sharing and accessing agriculture risk data which ultimately reduced the processing delays in insurance payouts. Another enchanting example is a privacy-preserving vehicular data sharing framework based on blockchain having features of tamper-proof, traceability, and decentralization (Huang et al., 2023). It employs an anonymous and auditable data sharing method based on Zero-Knowledge Proof (ZKP) technology to ensure vehicle identity privacy while keeping the auditability of vehicular data for Trusted Authorities (TAs). Another fascinating example is flight operation data sharing, which brings enormous benefits to all participants and improves data security by putting forward higher requirements (Xu et al., 2023). The flight operation data sharing uses blockchain technology, which improves trust, privacy and security, and transparency due to its decentralized structure. Last but not least, the case of smart cities is another intriguing example of data sharing where data sharing is compulsory to enable smart city components such as smart mobility and smart environment (Pradhan & Singh, 2021; Savithramma et al., 2022). Smart cities act as both data providers and consumers. More data is being shared by the organizations in order to stimulate innovation and address macro-level concerns in public health, welfare, climate change, and rural settings such as farms.

Besides data sharing, another critical factor in the digitized world is the generation of huge volumes of data by IoT. This huge volume of data helps to understand social trends or needs in domains such as transportation, food, energy, retail, e-commerce, agriculture, and health care. This can be seen as a key motivator for savvy urban environments (Bibri & Krogstie, 2020; Hashem et al., 2016; Liu & Huang, 2019). This huge volume of data may come across from a variety of data sources generated by a large variety of participants such as network administrators, framework suppliers, end users, and institutional entertainers (Cao et al., 2015). Before the data gets to its destination, it may undergo a number of modifications or changes. Even while data sharing, a huge volume of data, and data across a variety of sources are crucial to the modern digital economy, there have been several privacy and data misuse concerns.

General Data Protection Regulation (GDPR) is considered the most significant shift in data privacy regulations over the last two decades (Zaeem & Barber, 2020). On May 25, 2018, the European Union (EU) GDPR began to take effect across all EU member states (Li et al., 2019). It addresses data protection regulations, which establish guidelines for the processing, storage, and management of personally identifiable data (PID) for EU people. Organizations encounter difficulties in binding GDPR rights with enterprises, particularly those that are involved in digital contracting services. Binding GDPR rights with enterprises offers a new challenge and opportunity for enterprises to manage PID. To comply with GDPR, organizations must implement security safeguards on their services and procedures. Any type of organization that wants to process PID must satisfy one of the six legal bases defined by GDPR (see Section 1.7.1). To handle PID, prior consent (i.e., legal basis) from the data subject (or data owner) must be obtained. However, there are scenarios where consent is not enough, the selling of data, for example, where a contract complies with GDPR is required (Tauqeer et al., 2022).

Contracts play a key role in the world's digital revolution. They have evolved into a crucial component of an organization's assets for establishing long-term competitive advantages, as they are incorporated into the organization's vision and strategy, as well as its global business culture (Algarni, 2021). Moreover, they have become one of the core competencies of any successful 21st-century business. Over the last decade, digital transformation has drawn more attention to the creation of automated contract processes in digital contracting services (Algarni, 2021; Li & Samavi, 2018). The main reason behind this is that in the paper-based environment, slow and manual contract processing has a detrimental impact on a company's capacity to deliver outstanding services. Legal, financial, sales and other professionals can use their skills more effectively with the help of automated, digital contract lifecycle management (CLM). The CLM consists of stages such as drafting, negotiation, formulation, and fulfillment (Algarni, 2021; Chieu et al., 2007; Hassan et al., 2023). Since the negotiation stage involves numerous iterations until the contractors agree on the precise contractual terms and conditions (consisting of contractual obligations), the contract writing and formation stages take a lot of time and work.

It is essential to safeguard the signing stage of CLM for contracts in addition to complying with GDPR (by utilizing digital signatures). To accomplish business objectives, these stages of the CLM need to be strengthened and safeguarded. Another crucial aspect regarding contracts is the complexity of terms and conditions which often contain multiple pages and offers by the service providers. Customers sometimes fail to read these terms and conditions because they are frequently lengthy. However, individuals must accept those contractual terms and conditions (often naively) to fully benefit from digital services. For instance, when clients accept offers for auto-renewal, cellphone firms retain credit card information. Customers frequently cannot directly delete the information from their systems once they have accepted these agreements. Transparently visualizing terms and conditions for clients is a huge difficulty in the digital era.

Further, a contract becomes fully lawful if it complies with GDPR. To comply fully with GDPR, contracts must fulfill all the requirements such as technical and organization measures (TOMs), data security and privacy, and data protection by design principles defined by GDPR. Moreover, following these requirements, GDPR also posed other additional requirements such as the responsibilities of the data controller (DC) and the data processor (DP) for data processing (see Section 1.7.1) that also need to be fulfilled. In addition, using contracts as a GDPR-legal basis for data sharing requires a lot of time and effort due to the complexity of CLM.

With the importance of contracts complying with GDPR for data sharing in the digitized world, digital assets licensing "to make licenses (automated clearance of rights) on digital assets" (Havur et al., 2018; Pellegrini et al., 2019) is another important factor highlighted in the last decade. A digital license "is an official permission or permit to do, use, or own something (as well as the document of that permission or permit)" [1], automated clearance of rights on digital assets, legally secures and utilizes third-party data sources. The difficulty of automated rights clearance in the commercial exploitation of derivative works [2] has increased due to the growing popularity, particularly of protective and permissive licenses (Pellegrini et al., 2018). Transaction costs [3] rise in the manual clearance of rights on digital assets. Digital licensing can decrease these transaction costs. Without digital licensing, it is impossible to reap the full benefits of data sharing on digital assets. The digital licensing administration is a time-consuming, difficult, and error-prone activity. The Data Licenses Clearance Center (DALICC) (Pellegrini et al., 2018) and the License Clearance

[1]https://en.wikipedia.org/wiki/License
[2]https://en.wikipedia.org/wiki/Derivative_work
[3]https://en.wikipedia.org/wiki/Transaction_cost

Tool (LCT) ("License Clearance Tool", 2021) are two common solutions for the automatic clearance of rights for digital assets. However, there is still a need to investigate existing methods for the deployment and maintenance of digital licenses on digital assets such as software, content, and applications. A careful comparison of the existing tools is required to determine the best alternative among these tools. Furthermore, digital assets licensing provides standard licenses such as the Massachusetts Institute of Technology (MIT) [4]. Licenses are not part of GDPR but have legal bases, included in the EU law [5] ("European Public Licence", 2017). Combining these standard licenses with contracts (i.e., comply with GDPR) can create another GDPR-legal basis (i.e., license based contracts) for data sharing. With the help of GDPR-compliant license based contracts, contractors not only use standard licenses but also create additional requirements for data sharing (or processing), which gives them more control over data. This is another motive for this research by enabling multiple GDPR-compliant legal bases such as contracts, consent based contracts, and license based contracts for data sharing.

In addition, by enabling multiple GDPR-compliant legal bases for data sharing, an important task is to perform Knowledge Validation (KV) to ensure the correctness, improvement, and quality of knowledge. The performance of personal assistants and search engines (e.g., Yandex, Google, and OpenStreetMap) relies on high-quality knowledge bases, in particular, knowledge graphs (KGs). The KV is an important task and is responsible for measuring the degree to which KGs' triples are semantically correct. Several approaches have been proposed for validating KGs (Gad-Elrab et al., 2019; Lehmann et al., 2012; Rula et al., 2019). Eventually, most of them focus on validating knowledge regarding the Web and Wikipedia. Further, as far as we are aware, there have not been any studies exploring the KV through the aggregation of corresponding instances from alternative weighted structured knowledge sources.

Furthermore, to improve data sharing in the smart cities, insurance, and healthcare domains, important tasks are data digitization and the semantic modeling of data to make them easily accessible to end-users, such as healthcare providers and patients. The digitization of healthcare has led to the development of various resources (e.g., software facilitating patient access to health information), enhancing healthcare services by improving their efficiency and cost-effectiveness (Kuo, 2011). This results in massive amounts of multidisciplinary health data, which makes it challenging to fully exploit it. On the other hand, semantic technologies such as ontologies and KGs are employed to transform information silos across the entire health system into more

[4]https://opensource.org/license/mit/
[5]https://ec.europa.eu/isa2/solutions/european-union-public-licence-eupl_en/

reusable data that are useful for various health-oriented applications (Kalaycı et al., 2020). In addition, the research community has shown some interest in Question-Answering over Knowledge Graphs (QAKG). A QAKG system can be employed to address queries and aid users in accessing pertinent and relevant information (Aghaei, 2021; Chakraborty et al., 2019). Utilizing formal query languages with precisely defined syntax, like SPARQL (Swathi et al., 2016) and GraphQL (Taelman et al., 2018), enables the retrieval of information stored within KGs. Yet, employing formal queries to access knowledge within KGs presents challenges for users who are not experts in the field. Users must comprehend both the syntax of the formal query language and the fundamental structure of the Knowledge Graph (Aghaei, 2021). On the contrary, a User Interface (UI) for QAKG aids end-users, such as patients and medical professionals, in asking everyday healthcare-related questions in natural language, using their own terminology, and obtaining succinct answers (Chakraborty et al., 2019). In order to assist end-users and ease the process of query creation, there is a need for a UI that empowers users to access, explore, visualize, and comprehend data and its interconnected relationships presented in a graphical structure, which is another motive of this research.

1.2 Research Objectives

This study aims to improve present data sharing and consumption practices in the digital economy. At the same time, the purpose is to ensure compliance with the existing GDPR regulations (see Section 1.7.1). The main goal of this research effort is to improve data sharing infrastructures through the use of semantic contracts (contracts with semantic technologies such as ontology, KGs, web ontology language (OWL)), consent based contracts, license based contracts, and digital assets licensing. Other research objectives (ROs) are: (1) the development of semantic models for and compatible with contracts and licenses based on ontologies, (2) the implementation of automatic contract compliance and verification (CCV) checks that comply with GDPR, (3) the improvement of automated clearance of rights (for example, on digital assets such as contract source code) with DALICC or LCT, (4) the improving and safeguarding the signing stages of CLM by implementing digital signatures, (5) the implementation of license based contracts for data sharing comply with GDPR, (6) the improvement in the CCV compliance checking by utilizing Shapes Constraint Language (SHACL (see detail in Section 1.7.2.7)) validation as it is more semantically compliant, supports integration, and makes the validation process extendable with ease, (7) The creation and improvement of UI for QAKG to narrows the gap between users' questions and SPARQL queries, and (8) the improvement in the KV on knowledge bases and personal assistants for checking the correctness, improvement, and quality of knowledge.

1.3 Research Question

The main research question (RQ) in light of the aforementioned issues, challenges, and ROs is formulated as follows:
RQ: How can data sharing be facilitated with semantic contracts, licenses and knowledge structures in a variety of domains (such as smart cities, insurance, health)? It is further divided into the following research sub-questions (RSQs).

- **RSQ1:** How can semantic contracts be modeled and improved using an ontology (see Section 1.7.2.5) as a data model? (Chapter 2, Chapter 3, and Chapter 4)
- **RSQ2:** How can the CCV comply with GDPR and the CLM be performed to improve digital contracting services? (Chapter 3, and Chapter 4)
- **RSQ3:** How can multi-legal-based GDPR-compliant data sharing be enabled by employing consent, contracts, and licenses with SHACL? (Chapter 4)
- **RSQ4:** How can the gap between users' questions and SPARQL queries over QAKG be narrowed by employing UI? (Chapter 5, Chapter 3)
- **RSQ5:** How can the KV on personal assistant and knowledge bases be improved for checking the correctness, improvement, and quality of knowledge? (Chapter 6)

1.4 Research Methodology

This section details an outline of the research methods that were followed in this thesis. The research methodology of this thesis uses the following steps:

- **Literature Review:** The first step is to identify and understand the problem, and what has been done previously in this particular research area. As a result, appropriate literature was studied to comprehend the problem and the work completed. The literature review was done by the scope of the work being done. For instance, in the case of data sharing, one of the objectives is to enable GDPR-compliant data sharing in domains like smart cities and insurance. The literature review has been conducted for enabling multiple GDPR-legal bases such as consent based contracts, contracts, and license based contracts. Similarly, the literature review has been conducted for the other ROs such as performing KV over search engines (e.g., Google, Yandex) and the creation of UI for QAKG to narrow the gap between non-expert users' questions and SPARQL queries. Since our approach is based on semantic technologies, therefore, the other part of the literature review was based on the semantic modeling of contracts and licenses. Further, the literature review was done on SHACL-based validation which is a more semantically compliant solution (in many situations) compared to ad-hoc solutions. Furthermore, for the requirements of this research, the documents describing the requirements of the smashHit (The smashHit project, 2020a) project, datset collected from the Smell and Taste Center in Hospital Gelderse Vallei, Ede, the Netherlands, and the Tirol Knowledge Graph[6] (TKG) were examined. The outcome of this investigation is depicted in Chapter 2, Chapter 3, Chapter 4, Chapter 5, and Chapter 6.

- **Design and Implementation:** Following the literature review and understanding the requirements of the project, the next step is to design the system architecture (or experiment) and perform the implementation. The use cases UC1 (The smashHit project, 2020b) and UC2 (The smashHit project, 2020b) from the smashHit project (The smashHit project, 2020a) are the primary resources for the system design and its implementation. To fulfill RSQ1, the outcome of the literature review, a need to build or extend the existing semantic models of contracts. In a similar fashion, to fulfill RSQ2, a GDPR-compliant tool the CCV is implemented for the detection of contract breaches automatically. The tool also extends semantic models of contracts. In addition, RSQ3 is answered by extending the CCV tool with digital assets licensing, the implementation of digital signatures, enabling license based contracts along with GDPR, and the improvement in the CCV compliance checking by utilizing SHACL validation. Similarly, RSQ4 is answered by two phases: (1) By extending and enriching the existing ontology Clinical Smell and Taste Disorders Ontology (CSTD [7]) and the creation of KGs. (2) By implementing a UI for questioning-answering over KGs. In addition, to answer the RSQ5, a graph validator was implemented to perform KV on knowledge bases and search engines. The functional requirements from the smashHit project use cases that influenced the system design and its implementation in the case of data sharing are

[6]https://graphdb.sti2.at/sparql
[7]https://bioportal.bioontology.org/ontologies/CSTD

outlined as follows: (1) the semantic web (SW) technology, such as KGs, SPARQL Protocol and RDF (Resource Description Framework) Query Language (SPARQL), and SHACL must be used in the system design and its implementation, (2) the system design and implementation must meet the scalability requirements according to industrial use case, (3) GDPR legal requirements, such as data minimization and security and privacy must be met by the system design and implementation, (4) the system must perform compliance verification checks and trigger notification in any case of contract breach, (5) the implementation of the tool must be compatible and allow for easy integration with other systems. Similarly, the functional requirements for the KV on knowledge bases and the search engines are as follows: (1) use of the SW technology, such as KGs, SPARQL, (2) validation of a KG by finding the same instances across different knowledge sources, comparing their features, and scoring them, and (3) the score ranges from 0 to 1, which indicates the degree to which an instance is semantically correct for the task at hand. Finally, the functional requirements for the QAKG are as follows: (1) use of the SW technology, such as KGs, SPARQL, (2) extending the existing ontology the CSTD, (3) the creation of smell and taste disorder KGs, (4) the creation of questions with the help of experts in the domain of health, and (5) a graphical interface for interacting the users who are not expert technically. Furthermore, the system must be adaptable to additional use scenarios.

- **Evaluation:** The designed and developed approaches and/or tools for the conducted research are derived from the projects' requirements and standard evaluation metrics, including recall, precision, and F1-score. Furthermore, the obtained results of the experiments are compared to the state-of-the-art. Evaluation results are presented in Chapter 2, Chapter 3, and Chapter 4.

1.5 Contributions

This section describes the core contributions of the conducted research based on RQ and RSQs discussed in Section 1.3. The contributions are listed as follows:

1. The first contribution is the creation and enrichment of new and existing models of contracts and licenses for the purpose of data sharing using SW technology (RSQ1). For this purpose, different ontologies such as Multi-Tier Contract Ontology (MTCO) (Kabilan & Johannesson, 2003) and Financial Industry Business Ontology (FIBO) (Petrova et al., 2017) which

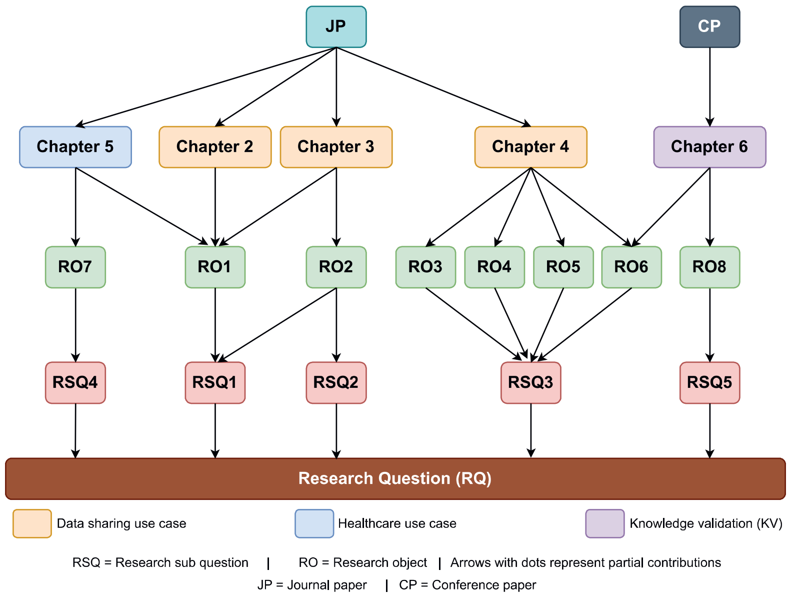
are the basis of our data sharing contracts have been studied. The initial semantic representation of the data sharing contracts can be seen in Section 2.5.2 while the extended version is in Section 3.3.1.

2. Our second contribution is to address GDPR-specific challenges regarding data sharing to improve digital contracting services through contracts (RSQ2). This is achieved by implementing a scalable and interoperable software tool. The software application not only addresses the challenges associated with GDPR but also supports scalable and interoperable processing and integration with other third-party software components. More details are provided in Chapter 3 and Chapter 4.
3. The third contribution is enabling multiple legal bases (i.e., consent based contracts, contracts, and license based contracts) for data sharing, improving and safeguarding the signing processes by implementing digital signatures in CLM (RSQ3). In addition, due to being more semantically compliant, SHACL is used for data validation instead of ad-hoc solutions. More details are described in Chapters, Chapter 3, Chapter 4.
4. Our fourth contribution is the enrichment and extension of the existing ontology CSTD and the creation of a smell and taste disorders KGs which is used to answer the questions over KGs (see details in Chapter 5).
5. Further, our fifth and final contribution is the creation of a graph validator application to perform KV on personal assistants and search engines (e.g., Google, Yandex) to ensure the quality of knowledge, which is measured by the degree to which statements are semantically corrected (details in Chapter 6).

Further, the relationship between RQ, RSQs, and chapters is presented in Figure 1.1. The most important UC is the data sharing in the smashHit project used by this thesis' Chapter 2, Chapter 3, and Chapter 4. The healthcare UC is used in Chapter 5, while Chapter 6 uses the KV UC. The Chapters, Chapter 2 and Chapter 3 contribute to RSQ1 and RSQ2, which are based on RO1 and RO2. Further, Chapter 4 contributes to RSQ3, which is based on the RO3, RO4, RO5, and RO6. In addition, Chapter 5 contributes to RO1 and R07, while Chapter 6 contributes to RO8 and R06.

1.6 Thesis Overview

This section details the structure of this cumulative thesis and provides a list of scientific publications in the form of chapters that contributed to it.



JP
CP
Chapter 5
Chapter 2
Chapter 3
Chapter 4
Chapter 6
RO7
RO1
RO2
RO3
RO4
RO5
RO6
RO8
RSQ4
RSQ1
RSQ2
RSQ3
RSQ5
Research Question (RQ)
Data sharing use case
Healthcare use case
Knowledge validation (KV)
RSQ = Research sub question | RO = Research object | Arrows with dots represent partial contributions
JP = Journal paper | CP = Conference paper

Figure 1.1: An overview of contribution regarding RQ, SRQ, and chapters.

1. **Chapter 2:** This chapter describes the smashHitCore (Kurteva et al., 2023) ontology based on two UCs: UC1 "Insurance Services" (The smashHit project, 2020b) and UC2 "Smart City Services" (The smashHit project, 2020c) in the smart cities and insurance domains. The ontology is enriched from different ontologies such as Gconsent [8], FIBO (Petrova et al., 2017), and Semantic Sensor Network Ontology (SSNO) [9]. The smashHitCore ontology provides semantic models for consent, contracts, licenses, and sensor data. This chapter corresponds to the publication "The smashHitCore Ontology for GDPR-Compliant Sensor Data Sharing in Smart Cities (Kurteva et al., 2023)" published in the Sensors journal. Further details are provided in Chapter 2.

2. **Chapter 3:** A GDPR-compliant tool - the CCV (Tauqeer et al., 2022) is implemented to perform digital contract management and contract compliance verifications checks comply with GDPR using semantic technologies. It supports different categories of contracts such as a contract based on consent, a Business-to-Consumer (B2C) contract, and a Business-to-Business (B2B) contract. Further, it enables data sharing between the Data Subject (DS) and the Data Controller (DC)/ Data Processor (DP) in

[8]https://openscience.adaptcentre.ie/ontologies/GConsent/docs/ontology
[9]https://www.w3.org/TR/vocab-ssn/

compliance with GDPR where consent is not enough. The main functionalities of the CCV tool are: (1) a scalable tool for managing semantic-based "digital contract data model using semantic technologies" contracts within the smart city and insurance UCs, (2) the tool implements a KGs-based approach for GDPR-compliant CCV, and (3) an ontology (i.e., used as a data model) and KGs for contracts that can be reused in various cases and domains. This chapter corresponds to the publication "Automated GDPR Contract Compliance Verification Using Knowledge Graphs (Tauqeer et al., 2022)" published in the Information journal. Further details are provided in Chapter 3.

3. **Chapter 4:** This chapter outlines multiple legal bases such as contracts, consent based contracts, and license based contracts fully compliant with GDPR for data sharing in smart cities and insurance domains. This work is extended to our previous work, the CCV (Tauqeer et al., 2022), which enables GDPR-compliant data sharing through consent and contracts only. The extended work includes: (1) the addition of license based contracts, another GDPR-compliant legal base, (2) making our previous work more semantically compliant by utilizing SHACL validation for compliance checking, (3) securing the CLM signing stage with digital signatures, and (4) the digital assets licensing. Further details are provided in Chapter 4.

4. **Chapter 5:** This chapter details a data-driven approach for Question Answering (QA) over smell and taste disorders KG (Tauqeer et al., 2023), which is based on the extended version of CSTD ontology. The real patients' data from a hospital in the Netherlands is used and transformed (using a semi-automatically method) into KG. Further, it supports QA in natural language and with SPARQL queries. In addition, a UI is developed for QA over KGs. Further details and implementation are discussed in Chapter 4. This chapter showcases creation and enrichment of existing ontology and the creation of KGs based on enriched ontology, ensuring creation of a semantic knowledge base for data sharing in further domains.

5. **Chapter 6:** This chapter outlines a graph validator application to perform KV on personal assistants and search engines (e.g., Google, Yandex) to ensure the quality of knowledge, which is measured by the degree to which statements are semantically correct (Huaman et al., 2021). It computes a confidence score for each triple and instance in KGs. A weight is assigned to each weighted source (i.e., Google, Yandex). The determined score is based on discovering the same cases across different weighted knowledge sources and comparing their attributes/features (e.g., name, phone).

The corresponding publication for this chapter is “Towards Knowledge Graphs Validation through Weighted Knowledge Sources” and presented in Iberoamerican Knowledge Graphs and Semantic Web Conference. This chapter showcases a possible approach to perform KV using KGs and ensuring appropriate data quality.

1.7 Preliminaries

In this section, we briefly introduce the core concepts and technologies used in our research. Section 1.7.1 details about GDPR while in Section 1.7.2, we give an overview of the semantic technologies.

1.7.1 GDPR

The EU GDPR (Li et al., 2019; Mondschein & Monda, 2019; “European Parliament and Council”, 2016; “GDPR”, 2018) entered into force on May 25, 2018, across all EU member states and replaced Data Protection Directive (DPD) (DATA, 1995). DPD is a directive for the data protection of individuals with regard to the processing of personal data and in effect since 1995. The GDPR, with 99 articles and 173 recitals, is one of the most extremely strict data protection legislation ever enacted to protect personal data. The following six legal bases are defined by GDPR (“European Parliament and Council”, 2016): (1) informed consent, (2) performance of a contract, (3) legal obligation, (4) protection of vital interests of the data subject, (5) performance of tasks carried out in the public interest or in the exercise of official authority vested in the controller, and (6) legitimate interest pursued by a DC. For lawful data processing, one of the aforementioned GDPR-legal bases must be met. The GDPR concerns everyone who collects and processes the EU citizens data (Mangini et al., 2020). As this research is mainly focused on data sharing based on consent, contracts, and licenses, therefore, we continue our discussion with two GDPR-legal bases regarding consent and the performance of contracts. However, prior to discussing both legal bases (i.e., consent and contracts), let us review some key terminologies that are used in GDPR and required for data sharing:

- **DS** A DS under GDPR (Art. 4(1)) is defined as “an identified or identifiable natural person” (GDPR, 2018d). A natural person is any person who wants to share data.
- **Personal Data** According to GDPR (Art. 4(1)) personal data is defined as “any information relating to an identified or identifiable natural person

('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person" (GDPR, 2018d).

- **Processing** Processing under GDPR (Art. 4(2)) is defined as "any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organization, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction" (GDPR, 2018d).

- **DC** The controller (or data controller in our case) according to GDPR (Art. 4(7)) is defined as "the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data; where the purposes and means of such processing are determined by Union or Member State law, the controller or the specific criteria for its nomination may be provided for by Union or Member State law" (GDPR, 2018d). In addition, GDPR also introduced additional responsibilities for the DC while processing personal data. The DC is responsible for and must be able to demonstrate compliance with, the principles governing data processing (according to GDPR (Art. 5) (GDPR, 2018e)). These principles include lawfulness, data minimization, accuracy, storage limitation and integrity, and confidentiality of personal data. Another responsibility of the DC is mentioned in GDPR (Art. 24) as "Taking into account the nature, scope, context and purposes of processing as well as the risks of varying likelihood and severity for the rights and freedoms of natural persons, the controller shall implement appropriate technical and organizational measures to ensure and to be able to demonstrate that processing is performed in accordance with this Regulation. Those measures shall be reviewed and updated where necessary" (GDPR, 2018a).

- **DP** The processor (or data processor) according to GDPR (Art. 28) is defined as "a natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller" (GDPR, 2018b). Further, similar to the DC responsibilities, GDPR also introduced the duties of the DP as described under GDPR (Art. 28): "Where processing is to be carried out on behalf of a controller, the controller shall use only processors providing sufficient guarantees to implement appropriate technical and organizational measures in such a manner that processing will meet the requirements of this Regulation and ensure the protection of the rights of the data subject" (GDPR, 2018b). In addition, in the case where the DC or the DP fails to comply with GDPR, they will be fined according to GDPR (Art. 83), which is defined as "the degree of responsibility of the controller or processor taking into account technical and organizational measures implemented by them" (GDPR, 2018h).

- **Consent** Consent under GDPR (Art. 4(11)) is defined as "any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her" (GDPR, 2018d). Further, GDPR introduced the following additional requirements for the fulfillment of consent: (1) **Informed** (the DS must be aware of both the purpose of processing and the identity of DC (Rec. 42) (GDPR, 2018i)), (2) **Unambiguous** (according to GDPR (Art. 7, Rec. 42) the information should be presented in an understandable and clear language without any unfair terms) (GDPR, 2018g, 2018i), (3) **Freely given** (the DS must have options to revoke or refuse consent and also separate consent option for different data processing (Art. 7, Rec. 42 and 43)) (GDPR, 2018g, 2018i, 2018j), and (4) **Specific** (according to GDPR (Art. 6 and 7) consent should be obtained for the use of specific data for a particular purpose) (GDPR, 2018f, 2018g).

- **Contractual performance** Recognizing the fundamentals of managing business operations (i.e., the obligations of a contract) is represented as another GDPR-legal basis according to GDPR (Art. 6(1)(b), Rec. 44) (GDPR, 2018f, 2018k) that allows data processing in two scenarios: (1) the data processing is permitted if it is necessary for entering into a new contract or working under an existing contract, and (2) the data processing is also permitted when the DS requests and initiate activities with the DC before entering into a contract. A typical example is the processing of credit card details in order to perform payment before entering into a contract. Similarly, another fascinating example could be when the DS makes an individual request for information from the service provider for a particular service via email or social network and the contract does not exist yet. Therefore, data processing is permitted for the purpose of responding inquiry.

In addition, for lawful data processing along with GDPR-legal basis consent, both the DS and the DC must comply with all consent criteria. Further, all the processing activities between the DS and the DC must be tracked in order to illustrate compliance and make all the records available to the higher authority upon request (Rec. 82) (GDPR, 2018l). Moreover, while keeping processing details records, GDPR requires the additional requirement from the DC or the DP to implement Technical Organization Measures (TOMs) such as encryption, to ensure the security of processing (Art. 32) (GDPR, 2018c).

1.7.2 Semantic Web

The term "Semantic Web" (SW) was introduced by Sir Tim Berners-Lee in 2001 and defined as "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better-enabling computers and people to work in cooperation" (Berners-Lee et al., 2001; Domingue et al., 2011). It enables the structuring and meaning of the information described in web pages and makes it possible machine-readable through a set of standards and technologies, such as ontologies, data exchangeable formats, and query languages. In the following, first an overview of semantic technologies (Sabou, 2016) is presented, and then a brief introduction to KG empowered by semantic technologies is described.

SW Technology SW technologies empower the integration and making sense of large-scale, evolving, and heterogeneous datasets (i.e., terms, concepts, and models) produced by various disciplines such as engineering, health, smart cities, and insurance (Sabou et al., 2017). This thesis relies on Resource Description Framework (RDF), RDF schema (RDFS), Simple Protocol and Resource Description Framework Query Language (SPARQL), Shapes Constraint Language (SHACL), ontologies, and Web Ontology Language (OWL), which constitute the base of the SW technologies.

1.7.2.1 RDF

Eric Miller (Miller, 2001) defines RDF as "RDF is an infrastructure that enables the encoding, exchange, and reuse of structured metadata. It is an application of eXtensible Markup Language (XML) that imposes needed structural constraints to provide unambiguous methods for expressing semantics." RDF is part of The World Wide Web Consortium (W3C) metadata activity that aims to produce a language for the exchange of machine-understandable descriptions of the information on the web (Brickley et al., 1998). In addition, it also enables metadata interoperability by designing mechanisms to support common conventions of semantics, syntax, and structure (Miller, 2001). RDF exchanges and processes metadata using XML as a standard syntax. Further, it allows for the publication of both human-readable and machine-readable vocabularies: "A set of properties, or structured data elements, defined by various communities (Miller, 2001)". RDF describes resources with statements (i.e., semantic triple [10] or fact). A semantic triple consists of three parts: (1) the subject, (2) the predicate, and (3) the object. Both the subjects and predicates are Uniform Resource Identifiers (URIs) while the objects are either URIs or literals (e.g., string, int, dateTime). The predicates connect the subject and the object via relationships. Let us take an example "Amar Tauqeer studies at WUR" from the natural language to demonstrate RDF. Using the semantic triple, "Amar Tauqeer" would correspond to the subject, "studies at" correspond to the predicate, and "WUR" would correspond to the object. Listing 1.1 shows the serialized formal RDF representation in Turtle (Beckett et al., 2014) for the aforementioned example. Other serialization formats of RDF are JSON-LD (Sporny et al., 2020) and N-Triples (Beckett, 2014).

Listing 1.1: A serialized formal representation of RDF in Turtle format.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix orcid: <http://orcid.org/> .

orcid:0000-0002-3345-387X dcterms:created "2022-12-22T22:25:56.900Z"
                                                                   ^^xsd:dateTime;
                        rdf:type foaf:Person;
                        rdfs:label "Amar Tauqeer";
                       :studiesAt <https://www.wur.nl/en.htm> .
```

1.7.2.2 RDFS

RDFS (W3C, 2004b) is a schema language and an extension of RDF (Allemang et al., 2020; McBride, 2023). It extends the basic RDF vocabulary for describing groups of related resources and the relationships among them. These resources are then used to determine other resources' characteristics such as domains and ranges of properties. An RDFS example is shown in Listing 1.2.

Listing 1.2: An example of RDFS.

[10]https://en.wikipedia.org/wiki/Semantic_triple

```
@prefix : <http://www.example.com/ontologies/smell-taste-disorder#> .
@base <http://www.example.com/ontologies/smell-taste-disorder> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

base:Patient rdf:type rdfs:Class ;
            rdfs:comment "Describes patient related information such as
                                                gender, length, height."@en;
            rdfs:label "Patient"@en;
            rdfs:subClassOf schema:Thing;

base:patientNo rdf:type owl:DatatypeProperty ,
                     owl:FunctionalProperty ;
              rdfs:domain base:Patient ;
              rdfs:range xsd:int ;
              rdfs:comment "Associate patient number to a patient."@en ;
              rdfs:label "patientNumber"@en .
```

A class “base:Patient” with a data property “base:patientNo” is defined using RDFS. The patient class uses “rdfs:subClassOf” RDF descriptor construct to describe the relationship with the class “schema:Thing”. On the other hand the data property “base:patientNo” uses “rdfs:domain”, “rdfs:range”, “rdfs:comment”, and “rdfs:label” RDF descriptor constructs for describing patient information.

1.7.2.3 OWL

In the previous Section 1.7.2.2, we demonstrated the RDFS provides additional descriptor constructs for RDF due to its limitation (e.g., how to define classes and properties with their relationships). Similarly, RDFS also has limitations to express the vocabulary. As an example, RDFS does not have descriptor constructs such as to differentiate between two concepts and determine whether the two classes are disjoint or not (Hogan, 2020). OWL supports more machine interpretability of web data that is supported by XML, RDF, RDFS (W3C, 2004b). To describe the classes and properties (e.g., symmetry) with addition constraints/restrictions, OWL provides a rich set of vocabulary and restrictions (Allemang et al., 2020; W3C, 2004b). In addition, OWL is an ontology language pertaining to formally defined meanings (W3C, 2004a).

1.7.2.4 SPARQL

SPARQL is a W3C recommendation and a structured query language for querying RDF data through a semantic triple or graph pattern (SPARQL Working Group, 2008; Swathi et al., 2016). It is used to define queries across several data sources, whether the data is stored natively as RDF or seen as RDF via middleware (W3C, 2013). It performs similar functionalities such as projection, join, union, and constraints with the role of Structured Query Language (SQL) as it relates to relational databases (SPARQL Working Group, 2008; Swathi et al., 2016). In addition, it supports aggregation, sub-queries, extensible value testing, and constraining queries by source RDF graph. For demonstration purpose, a SPARQL query sample is presented in Listing 1.3 (Tauqeer et al., 2023).

Listing 1.3: A sample of SPARQL query.

```
prefix : <http://example.com/base/>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select ?hasEtiology (count(?hasEtiology) as ?total)
where{
select *
where {
    ?patient a :Patient;
          :hasSmellDisorder ?smell;
          :hasEtiologyType ?hasEtiology .
     filter(?smell="anosmia")
}
}
group by ?hasEtiology
order by desc(?total)
```

The SPARQL query in Listing 1.3 returns the type of etiology along with totals in descending order of a patient of smell disorder “anosmia”. The result of this query can be found in Chapter 5, Figure 5.9.

1.7.2.5 Ontology

An ontology is a formal description of knowledge as (1) a set of concepts (also known as classes) within a domain (e.g., protein is a concept within the domain of molecular biology) and (2) the relationships among the concepts (e.g., an enzyme is a kind of protein, which in turn is a kind of macromolecule) (Gruninger & Lee, 2002; Tauqeer et al., 2023), Chapter 4. It serves as the focal point of any

SW-based solution, allowing for the explicit and formal description of knowledge pertinent to the application at hand. The semantic integration of data can be achieved through ontologies as mediators (Sabou, 2016). Similarly, properties are sometimes also called shared conceptualization” by Studer et al. (Studer et al., 1998), which is extended with two other definitions by Gruber (Gruber, 1993) and Borst (Borst, 1997). Here, on the one hand, the term “formal” refers to the syntax and semantics for enabling machine readability. While on the other hand, the term “explicit” refers to defining all the concepts and restrictions explicitly. In a similar way, the term “conceptualization” indicates real-world entities such as classes, and properties (i.e., object properties (define the relationships between classes) and data properties (define relationships between an entity instance and data value (e.g., integer, string))) must have an abstract representation. Another powerful feature of ontologies is to provide formalizations for machine-readability and reasoning capabilities by providing restrictions between classes and properties. A snippet of the “Smell and Taste Disorder Ontology” (Tauqeer et al., 2023) is shown in Figure 1.2. The “cstd:Patient” and “cstd:MedicationType” are two classes that represent the concepts associated with patients’ records as depicted in Figure 1.2. Further, the data properties “cstd:hasAntiAllegic” and “cstd:hasAntiBiotic” can aid additional constraints in the form of relationships between distinct classes and the types of values they should contain. An object property can be used to define a relationship between classes by establishing a connection through domain and range constraints. More details about ontologies and their semantic representations are presented in Chapter 2, Chapter 3, Chapter 4, and Chapter 5.

1.7.2.6 KGs

KGs, on the other hand, contain the actual data. Google coined the term KG in 2012 as the foundation of a new web search methodology (Fensel et al., 2020). A more concise and concrete definition of the KG is “a graph of data intended to accumulate and convey knowledge of the real-world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities” was published by Hogan et al. (Hogan et al., 2021).

Ontologies are semantic models that can only model general types of things but do not store information about individuals in a particular domain. A sample of the smell and taste disorders KGs is presented in Listing 1.4 while a graphical representation is shown in Figure 1.3.

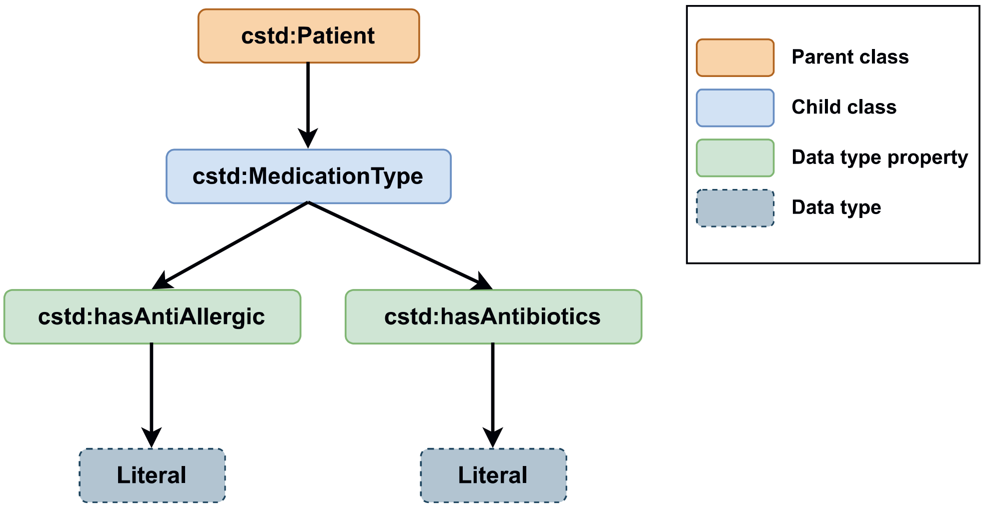


Figure 1.2: A snippet of smell and taste disorders ontology.

Listing 1.4: A sample of KG.

```
@prefix : <http://example.com/base/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix schema: <https://schema.org/> .

<http://example.com/base/118190> rdf:type owl:NamedIndividual ,
                                          base:Patient ;
                                  base:patientNo 118190 ;
                                  base:endoNasalScopy "Conchae hypertrophy" ;
                                  base:hasAge "56" ;
                                  base:hasBMI 27.78 ;
                                  base:hasComplaintDuration "2-5 years" ;
                                  base:hasDiagnostic "Both" ;
                                  base:hasEntTherapy "Nasal corticosteroids" ;
                                  base:hasEtiologyType "Idiopathic" ;
                                  base:hasLength 1.79 ;
                                  base:hasParosmia "2" ;
                                  base:hasRetronasalTest 7.0 ;
                                  base:hasSmellDisorder "anosmia" ;
                                  base:hasSmellTherapy "Smell_therapy" ;
                                  base:hasTasteDisorder "Normogeusia" ;
                                  base:hasWeight 89.0 ;
```

```
                              base:historyOfEntSurgery "Yes" ;
                              base:onsetOfComplaint "Gradual" ;
                              base:presenceOfComplaint "Constant" ;
                              base:sniffinStickDiscrimination 6.0 ;
                              base:sniffinStickIdentification 5.5 ;
                              base:sniffinStickTDI 12.5 ;
                              base:sniffinStickThreshold 1.0 ;
                              base:symptomsAtDiagnosic "Nasal obstruction" ;
                              base:tasteStrip 14.0 ;
                              schema:gender "Man" .
```

In addition, the KG contains information about a patient individual “118190” regarding gender, height, medication, and the type of medication.

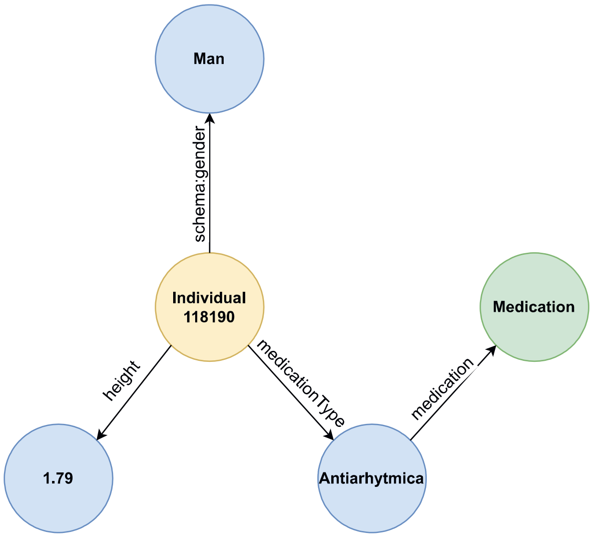


Figure 1.3: A sample of smell and taste disorders KG.

1.7.2.7 SHACL

SHACL, “a language for validating RDF graphs against a set of conditions” (Knublauch & Kontokostas, 2017), is a World Wide Web Consortium (W3C) recommendation for validating KGs. There are two types of graphs used in SHACL validation: a shape graph and an RDF graph. The former is used to contain the constraints or conditions and other constructs, which are expressed in the form of an RDF graph. While the latter is an RDF graph, which is validated against the SHACL shape graph. Further, there are two types of shapes defined in the SHACL core language: node shapes “A node shape is a shape in the shapes graph that is not the subject of a triple with sh:path as its predicate [11]” (Knublauch & Kontokostas, 2017) and property shapes “A property shape is a shape in the shapes graph that is the subject of a triple that has sh:path as its predicate” (Knublauch & Kontokostas, 2017). The examples of node shapes and property shapes are presented in Listing 1.5, while an example of an RDF graph is shown in Listing 4.1.

Listing 1.5: A sample of shape graph.

```
@prefix base: <http://ontologies.atb-bremen.de/smashHitCore#> .
@prefix fibo-fnd-agr-ctr: <https://spec.edmcouncil.org/fibo/
                                            ontology/FND/Agreements/Contracts/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema: <http://schema.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

schema:CCVThirdScenarioElsePartShape a sh:NodeShape ;
 sh:targetClass base:CCVThirdElse;
 sh:and (
        [a sh:NodeShape;
         sh:property [
              sh:path base:hasContractCategory;
              sh:in (base:categoryBusinessToConsumer)
                ]
        ]
        [a sh:NodeShape;
         sh:property [
              sh:path base:hasConsentState;
              sh:in ("Valid")
                ]
         ]
);
 sh:sparql [
 a sh:SPARQLConstraint ;
 sh:message "Violation: Contract end date has been passed." ;
 sh:select '''
    SELECT $this
    WHERE {
     $this base:hasStates ?state .
     $this base:hasEndDate ?edate .
     $this base:currentDateTime ?cdate .
```

[11]https://www.w3.org/2014/data-shapes/track/issues/220

```
     $this base:hasContractStatus ?cstatus .
    FILTER ((?state=base:statePending) && (?cdate > ?edate) && (?cstatus
           IN (base:statusCreated, base:statusUpdated, base:statusPending)))
    }
    ''' ;
  ] .
```

There are three node shapes (one outer node shape and two inner node shapes) and two property shapes used in Listing 1.5. The first inner node shape has a property shape, which restricts the contract with the “business to consumer” category. With this restriction, no other contract categories are allowed. While the second inner node shape has a property shape, which only allows the contract with a “Valid” consent state. Listing 4.1 shows an RDF graph containing the data in the form of semantic triple and can be validated against the shape graph described in Listing 1.5.

Listing 1.6: A sample of data graph.

```
@prefix base: <http://ontologies.atb-bremen.de/smashHitCore#> .
@prefix fibo-fnd-agr-ctr: <https://spec.edmcouncil.org/fibo/
                                        ontology/FND/Agreements/Contracts/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

 base:contb2c_f6172eaa-a2e6-11ed-b1bc-0b613fb8badf a base:CCVThirdElse;
    base:contractType base:Written;
    base:forPurpose "data sharing between processor and consumer";
    base:hasContractCategory base:categoryBusinessToConsumer;
    base:hasContractStatus base:statusUpdated;
    base:hasEndDate "2024-07-07T10:38:07"^^xsd:dateTime;
    fibo-fnd-agr-ctr:hasEffectiveDate "2023-02-02T14:14:25"^^xsd:dateTime;
    fibo-fnd-agr-ctr:hasExecutionDate "2023-02-02T14:14:25"^^xsd:dateTime;
    base:hasStates base:statePending;
    base:consentState "None";
    base:currentDateTime "2023-02-08T13:50:28"^^xsd:dateTime .
```

CHAPTER 2

The smashHitCore Ontology for GDPR-Compliant Sensor Data Sharing in Smart Cities

This chapter is based on:

Kurteva A., Chhetri TR., Tauqeer A., Hilscher R., Fensel A., Nagorny K., Correia A., Zilverberg A., Schestakov S., Funke T., Demidova E. The smashHitCore Ontology for GDPR-Compliant Sensor Data Sharing in Smart Cities. Sensors. 2023; 23(13):6188. https://doi.org/10.3390/s23136188 (Kurteva et al., 2023)

Abstract

The adoption of the General Data Protection Regulation (GDPR) has resulted in a significant shift in how the data of European Union citizens is handled. A variety of data sharing challenges in scenarios such as smart cities have arisen, especially when attempting to semantically represent GDPR legal bases, such as consent, contracts and the data types and specific sources related to them. Most of the existing ontologies that model GDPR focus mainly on consent. In order to represent other GDPR bases, such as contracts, multiple ontologies need to be simultaneously reused and combined, which can result in inconsistent and conflicting knowledge representation. To address this challenge, we present the smashHitCore ontology. smashHitCore provides a unified and coherent model for both consent and contracts, as well as the sensor data and data processing associated with them. The ontology was developed in response to real-world sensor data sharing use cases in the insurance and smart city domains. The ontology has been successfully utilised to enable GDPR-complaint data sharing in a connected car for insurance use cases and in a city feedback system as part of a smart city use case.

Keywords— ontology, consent, contracts, sensors, data sharing, GDPR, smart cities, insurance.

2.1 Introduction

The GDPR (“GDPR”, 2018), which came into effect in 2018, has introduced six legal bases, namely consent, contract, legal obligation, vital interest, public task and legitimate interests for the lawful processing of personal data (“Lawful basis for processing”, 2018). Consent is arguably the most widely used GDPR basis and specific requirements apply to it. It should be specific, unambiguous, freely given, and informed and should be implemented in a way that it can be withdrawn as easily as it was given (Art. 7, Rec. 32, 42, 43). However, depending on the type of data processing and context a contract might be needed instead (“Lawful basis for processing”, 2018). In comparison to consent, contracts have more restrictions in the form of terms and conditions with specific obligations (Art. 6(1)(b)), which depend on the nature of the contract (e.g., sale of goods, e-commerce, education, services) (Freire & de Valk, 2019).

In smart cities, one’s data is spread across different sources (such as vehicles sensors, smart home sensors, databases, people, etc.) and can be used for different purposes by different entities simultaneously. For example, an individual can report a damaged road with a photo and location information (e.g., GPS data) to authorities so that authorities can take appropriate action. The main challenge is the management of the consent itself as each entity (e.g., company) might store it in different formats and locations. Locating and analyzing all consent information is time-consuming and still a challenge for many small and medium enterprises (SMEs) as discussed in (The smashHit project, 2021). Insurance companies handle personal data on a daily basis, making them one of the industries highly impacted by the GDPR. One of the main challenges when modeling consent for insurance companies is the existence of dynamic situations such as a change in ownership of the insured property (e.g., a car), which can disrupt existing declarations of consent. In our study, we have considered two sensor data sharing use cases, where GDPR compliance is expected. Use case one (UC1) focuses on vehicle sensor data sharing in smart cities, while use case two (UC2) focuses on vehicle sensor data sharing in the insurance domain. Details about each use case are presented in more detail in Section 2.2.

Semantic Web technologies, such as ontologies, can represent diverse dynamic contexts and support faster and easier knowledge discovery, data interoperability, and transparency (Freire & de Valk, 2019; Lakehal et al., 2019; Murtazina & Avdeenko, 2019). Further, ontologies help translate human knowledge into a machine-understandable format to be used as a knowledge management infrastructure within computer systems (Fensel, 2001). These capabilities of ontologies make them a well-fitting solution for the representation and interpretation of the heterogeneous data in UC1 and UC2. As presented later in Section 2.3, several semantic models for consent, contracts and sensors have already been built. However, our analysis in (Kurteva et al., 2021) and the requirements of UC1 and UC2 point to three areas of improvements that require an addition to the set of existing ontologies. First, the concepts of consent and contract are treated as exclusive with ontologies modelling either consent or contract.

Second, specific data sources such as sensor data are in most cases not modelled in detail (e.g., what specific sensor is the source of the data). Furthermore, finally, not all ontologies are openly available. Consent for data processing should be requested in an informed manner. Individuals need to be presented with information about the type of data that will be processed and the specific purpose of the processing (e.g., GDPR Art. 4, 6, 7 and Rec. 32, 42). For example, consent for vehicle sensor data sharing is an ambiguous request as a vehicle has multiple sensors for different observations. To be compliant with the GDPR, one should know exactly the type of sensor and the corresponding data the consent covers (e.g., GPS sensor data and more specifically latitude and longitude data). Third, existing ontologies are limited when it comes to modelling specific concepts needed for complex scenarios such as broken consent chains (e.g., due to the transfer of property ownership (Chhetri et al., 2022b)).

In our UC1 and UC2 it is also possible that a contract is the adopted GDPR legal basis for data sharing and processing. This can be due to (i) data subjects being presented with a contract instead of a consent request (depending on the company) and (ii) data subjects interested in negotiating the terms and conditions for the data processing. When dealing with consent the obligations for the data subject are pre-defined, while contracts allow their negotiation. In UC1 and UC2, we have considered the possibility of both business to customer (B2C) and business to business (B2B) contracts, where the involved contractors can negotiate specific terms and conditions for data processing. Data selling between companies is a typical scenario where a contract is required (and consent is not enough). Another difference is that individuals have the right to revoke their consent at anytime, while contracts have specific steps that need to be followed for cases such as contract termination. To summarise, the use cases that motivated our work consider two GDPR legal bases for (sensor) data sharing—consent and contracts. To our knowledge, an ontology that consistently represents both concepts, dependencies between them and the relevant sensor data, has not yet been built.

Based on the need for such an ontology highlighted by smashHit's (The smashHit project, 2020a) UC1 and UC2 and the GDPR (Art.6, Rec. 32, Rec. 44), we present the smashHitCore Web Ontology Language (OWL) (W3C, 2004c) ontology. The ontology and its accompanying documentation are publicly accessible at (The smashHit project, 2022).

smashHitCore, which is openly available, reuses and extends existing ontologies to provide a harmonised model for consent, contracts, sensor data and its processing. smashHitCore can be used in simpler cases such as informed consent modelling based on GDPR or more complex ones such as automatic contracting (i.e., automatic contract generation, signing and execution) and automated compliance verification based on one's informed consent (see (Chhetri et al., 2022b; Tauqeer et al., 2022)).

The rest of the paper is structured as follows. The use cases that motivated our work are presented in Section 2.2. Section 2.3 presents an overview of existing ontologies for consent, contracts and sensor data, while Section 2.4 presents the methodology we followed for the development of smashHitCore. The ontology specification is presented in Section 2.5. The specific applications of smashHitCore for consent and contract compliance is presented in Section 2.6. Section 2.7 presents the evaluation of smashHitCore. Conclusions can be found in Section 6.6.

2.2 Use Cases

The smashHitCore ontology is based on two sensor data sharing use cases. UC1-Smart City Services is presented in Section 2.2.1, while UC2-Insurance Services is presented in Section 2.2.2

2.2.1 UC1-Smart City Services

The increasing volume of smart city sensor data about traffic, road conditions, carbon emissions and the vehicles contributing to all these, holds great potential to be utilised for optimising current mobility services and for implementing more efficient ones in the future (The smashHit project, 2020c). Helsinki is an example of a leading smart city, which has set as its goal to become one of the most functional cities in the world. For Helsinki, functionality means above all ensuring convenient everyday life for its citizens, visitors, and businesses by fully harnessing the potential of digitization. The aim of the city is to enhance the utilization of data and analytics, to produce individually tailored services for the city residents, proactively and when they need them. The city is committed to the MyData (MyData Global and KIRAHub, 2022) principle, according to which residents must be able to manage the personal data that the city collects (e.g., authorise its use for different purposes and services, be able to restrict and even deny access to it). For example, personal data (e.g., vehicle sensor data such as fuel, speed, location) is essential for the current traffic planning and management digital services. Fluency and safety in city traffic have an enormous influence on the well-being of the residents, visitors, and businesses. Optimizing traffic flows and enhancing safety can only be conducted through a joint effort of the city, its residents, visitors, and businesses. However, such vehicle sensor data are considered personal, which means it should be shared and processed in a GDPR-compliant way that does not endanger one's privacy. To be GDPR-compliant, informed consent needs to be requested, received and managed through its whole lifecycle (Kurteva et al., 2021). The main challenge is to perform all of the above for the heterogeneous sensor data at scale while complying with the GDPR.

2.2.2 UC2-Insurance Services

UC2 focuses on insurance services that also rely on vehicle sensor data sharing (The smashHit project, 2020b). With the growing ability of vehicles to generate, gather and share car data with third parties among different data sharing platforms, there is a need for flexible and easily manageable procedures to handle data owner consent and legal compliance, to achieve effective and traceable contracting. Current fleets of connected vehicles cannot provide the data required to support advanced usage-based insurance models without additional intervention during the workflow between vehicles and insurers. The challenges and complexities of the GDPR directives make cumbersome mechanisms necessary to gain, record, and manage consent and contract. The concerns of personal data misuse are also rising. The combination of understanding and relating to the value proposition, individuals' trust, and cumbersome consent and contract processes result in a low opt-in rate for connected vehicle data exchange and particularly for connected insurance programs. Another concern is the possibility of sensitive data (e.g., GPS location, driving trends) leakage, which can be a major threat to both Original Equipment Manufacturers (OEMs) and data processors alike. Preserving individuals' privacy and using their data in a GDPR-compliant way is a necessity.

2.3 Related Work

This section presents an overview of existing ontologies that model consent as defined by GDPR, contracts and the data related to them, specifically sensor data.

2.3.1 Semantic Models for Consent

This section is based on our previously conducted survey (Kurteva et al., 2021) on GDPR consent ontologies. One of the earliest ontologies that semantically represents consent as defined by the GDPR is the Consent and Data Management Model (CDMM) by Fatema et al. (Fatema et al., 2017). CDMM, available since 2017, represents consent as an entity within the context of a privacy policy. The ontology is able to represent consent through its whole life-cycle (i.e., from its request to its withdrawal). However, CDMM does not model meta-information such as data storage location, contact information of entities such as data controller, third parties and their obligations, which can be helpful when performing GDPR compliance verification.

Pandit et al. (Pandit et al., 2019a) present a GDPR-compliant consent ontology called GConsent (Pandit et al., 2019a). The ontology's main focus is to model consent-related information as defined by GDPR (Pandit et al., 2019a). Consent is semantically represented as an artefact, that has multiple states such as given, refused, revoked and unknown. In comparison to CDMM, GConsent focuses on GDPR compliance and models consent in more detail. However, it does not model specific responsibilities required for our use cases such as the entity responsible for consent, specific contact information and obligations regarding the use and revocation of consent.

The PrOnto ontology (Palmirani et al., 2018) focuses on modelling GDPR obligations and requirements such as consent for personal data processing. Further, PrOnto models the specific documents, data, actors involved in different data processing and the legal rules that can apply. However, PrOnto is not openly available and its proprietary status prevents a wider adoption by researchers and industry practitioners.

The Legal Compliant Ontology to Preserve Privacy for the Internet of Things (LloPY) (Loukil et al., 2018) is an ontology that models GDPR concepts, such as consent, but is not limited to it. The ontology further reuses privacy definitions provided by the National Institute of Standards and Technology (NIST) ("NIST", 2022). LloPY focuses on GDPR-compliant consent modelling for cases such as sensor data sharing. It models consent decisions, retention, disclosure, as well as types of sensor data by reusing the Semantic Sensor Network ontology (SSN) (Compton et al., 2012). Despite its potential to model diverse data, LloPY is not openly available, which is an obstacle to its wider evaluation and reuse by the legal and Semantic Web communities.

The Business Process Re-engineering and Functional Toolkit for GDPR Compliance (BPR4GDPR) (The BPR4GDPR project, 2019) represents consent, its states (e.g., provided, refused, revoked) and GDPR-related information that is needed for performing compliance checking. Based on its specification in (The BPR4GDPR project, 2019), BPR4GDPR models specific roles, event types, context types and state types related to consent. However, BPR4GDPR is not open-access.

Consent is modelled as a policy and is used for privacy compliance checking by both SPECIAL's Usage Policy Language (SPL) (Kirrane et al., 2020) and SPECIAL Policy Log Vocabulary (SPLog), which were built for the SPECIAL project [1]. The focus of both SPL and SPLog is to represent data usage policies in a machine-readable format and to define permissions based on one's consent decision (Kirrane et al., 2020).

The Data Privacy Vocabulary (DPV) (Pandit et al., 2019b) models consent and its attributes (e.g., expiry time) in the context of privacy policies, which can be used for compliance checking. DPV models other GDPR-related concepts such as notice, expiry date and provision. The ontology has been extended with specific knowledge about GDPR's legal basis, technical and organisational measures and processing categories (also known as DPV-GDPR ("DPV-GDPR", 2022). A current limitation, with regards to GDPR and consent, is that DPV does not model the responsibilities of data controllers (Kurteva et al., 2021).

2.3.2 Semantic Models for Sensor Data

Russomanno et al. (Russomanno et al., 2005) present an approach for constructing a sensor ontology that is a formal conceptualisation of sensors. The presented OntoSensor (Russomanno et al., 2005) ontology is based on the Sensor Model Language (SensorML) (Botts & Robin, 2007) and the Suggested Upper Merged Ontology (SUMO) (Niles & Pease, 2001) and models sensors' attributes, performance, capabilities and reliability. However, it has a limited expressivity with regard to describing the sensor's

[1] https://specialprivacy.eu/

observations, which are of significant importance for our UC1 and UC2.

Another widely used sensor ontology is SSN (Compton et al., 2012), a standard W3C recommendation in the field. SSN describes sensors, their observations, the involved procedures, the studied features of interest, the samples used to do so, the feature's properties being observed or sampled, as well as actuators and the activities they trigger (Haller et al., 2019). The ontology has been reduced by horizontal and vertical modularisation, which allows different users to only use domains that they are interested in.

The Sensor, Observation, Sample, and Actuator (SOSA) (Janowicz et al., 2019) ontology, which is the lightweight version of SSN, is event-centric. It focuses on modelling observations, sampling, actuations and procedures of sensors. While focusing on the interactions between sensors, and being easy to understand by a broader audience, the ontology does not provide the needed level of granularity of the sensors' themselves (e.g., specific types of sensors such as GPS). However, SOSA can still be reused for cases where sensor observations are the main focus.

The Ontonym-Sensor (Stevenson et al., 2009) ontology provides a high-level description of sensors and capabilities such as their frequency, coverage, accuracy, and precision pairs. In addition, sensor observations (observation-specific information, metadata, sensor, timestamp, and the time period over which the value is valid, the rate of change) are modelled as well (Stevenson et al., 2009). Due to its generic nature, Ontonym-Sensor can be reused in different scenarios that involve both sensors and sensor data.

Last but not least, Eid et al. (Eid et al., 2007) proposed the Sensor Data Ontology (SDO), which reuses the Suggested Upper Merged Ontology (SUMO) (Niles & Pease, 2001) for general definitions and associations and follows IEEE 1451.4 (Licht, 2001) (a standard for describing smart transducers and their observations). Although Eid et al. present an ontology, the main focus of their work in (Eid et al., 2007) is on sensor data search optimisation. Due to the limited information available, SDO can be viewed as lightweight foundational semantic model for sensors that can be built upon.

2.3.3 Semantic Models for Contracts

The Multi-Tier Contract Ontology (MTCO) by Kabilan and Johannesson (Kabilan & Johannesson, 2003) comprises three layers. The authors defined the conceptual models of contracts in the first layer. The specific contract types are defined in the second layer. The third layer is responsible for capturing contractual obligations and their fulfilment patterns. Furthermore, the proposed ontology contains different stages, such as drafting, negotiation, and signing, which can be useful in modelling semantic contracts. The performance obligations, rules, rights, and payments are additional contract details added by MTCO. The ontology does not, however, clearly distinguish between an electronic contract and a conventional contract.

In (de Cesare & Geerts, 2012), the authors propose a contract ontology, which defines three building blocks, such as agreements amongst persons, promises, and considerations to support modelling semantic contracts. The authors describe different types of contracts (e.g., verbal and written), and events related to the execution, fulfilment, and exchange of contracts. The formalisation of the ontology in OWL and the modelling of specific contract domains (e.g., sales), however, are left for future work. The ontology also predates GDPR, therefore its particular legal requirements for data processing have not been taken into account.

The Financial Industry Business Ontology (FIBO) (Bennett, 2013; EDM Council, 2020; Petrova et al., 2017), which is a collection of eleven distinct ontologies that define entities and processes in the business and finance domains, is a more recent ontology for contracts, while FIBO does not concentrate on particular laws such as the GDPR, it does offer a thorough semantic model of concepts like contracts and agreements that can serve as the basis for any ontology focusing on GDPR. Although FIBO does not explicitly consider GDPR when modelling contracts, there have been recent updates made regarding how the law is mapped out.

2.3.4 Summary

Based on our previous ontology survey in (Kurteva et al., 2021) and the related work overviewed in Section 2.3, we have identified several ontologies that can represent consent as a GDPR basis for lawful data processing. GConsent focuses specifically on representing informed consent as defined by GDPR but does not model specific sensors and sensor data, which are needed in cases such as UC1 and UC2 (see Section 2.1). CDMM, SPL and DPV focus on privacy policies. PrOnto (Palmirani et al., 2018) and Llopy (Loukil et al., 2018) model GDPR-compliant consent, however, both are not open access, which limits their reuse. Further, the ontologies that model GDPR focus exclusively on the concept of consent. Contracts, which are also a GDPR basis for data processing, are rarely modelled by consent ontologies. Similarly, the ontologies that model contracts (e.g., FIBO), do not model consent and do not follow a specific law such as GDPR. Sensors are rarely represented in consent and contract ontologies as a type of data source. The wide availability of ontologies for consent and contracts makes their reuse complex and time-consuming as it is not clear, which ontology to reuse, when, and how exactly.

2.4 Methodology

The smashHitCore ontology is a result of a collaborative effort of individuals from various backgrounds (i.e., Semantic Web, Privacy and Security, Law, Insurance and Mobility). We followed the methodology for ontology development by Noy and McGuiness (Noy

& McGuinness, 2001) and used the VocBench [2] environment for collaborative ontology development. The development of smashHitCore (The smashHit project, 2022) follows a hybrid approach, which is a combination of the ontology modelling guidelines presented by Uschold et al. (Uschold & King, 1995) and Noy (Noy & McGuinness, 2001). The following steps were carried out to build the ontology:

1. Use case identification and analysis of the range of intended users and specific use-case related requirements (i.e., competency questions) (see Table 2.1) with industry collaborators.
2. Identification of the main concepts needed for consent and contracts in UC1 and UC2. Review of GDPR requirements for informed consent with legal experts.
3. Gathering and analysis of existing ontologies for consent (see (Kurteva et al., 2021)), contracts, sensor data and data processing.
4. Creation of the smashHitCore ontology by reusing specific classes and their respective properties from existing ontologies. Definition of new use case-specific concepts.
5. Preservation of consistency by following the "isA" relationship between subclasses and classes (e.g., Sensor isA Device). Capitalisation of each word when labelling classes (e.g., Data Source) and following the British English language standard.
6. Evaluation of the expressivity of the ontology with the competency questions derived during Step 1 (see Table 2.1).
7. Evaluation of the correctness of the ontology with the HermiT (Shearer et al., 2008) reasoner.
8. Reviewing and editing of the ontology again if needed.

Table 2.1: Evaluation of smashHitCore with competency questions.

Questions	**Relevant smashHitCore concepts and object properties**
Informed consent questions:	
What is the purpose of the consent?	*gconsent:Consent, dpv:Purpose, dpv:hasPurpose, smashHitCore:hasContext, smashHitCore:atLocation, smashHitCore:inMedium, smashHitCore:isAboutData*
What is the duration of the given consent?	*smashHitCore:consentID, smashHitCore:atDateTime, smashHitCore:hasExpirationDate, smashHitCore:hasRetentionDate*
	Continued on next page

[2]https://vocbench.uniroma2.it/

Table 2.1 – continued from previous page

Questions	**Relevant smashHitCore concepts and object properties**
What is the status of consent?	*smashHitCore:consentID, gconsent:StatusInvalidFofProcessing, gconsent:Withdrawn, smashHitCore:Granted, gconsent:isStatusFor, gconsent:hasStatus*
What is the context of consent?	*smashHitCore:consentID, smashHitCore:Context, smashHitCore:hasContext, smashHitCore:atLocation, smashHitCore:atDateTime, smashHitCore:inMedium*
How was consent requested, granted, revoked?	*smashHitCore:consentID, gconsent:Consent, gconsent:Medium, smashHitCore:inMedium, consent:AppBased, consent:Audio, consent:Video*
Who provides the consent?	*gconsent:Consent, smashHitCore:consentID, prov:Agent, smashHitCore:agentID dcat:Role, smashHitCore:DataSubject, smashHitCore:isProvidedBy*
Data source and data processing questions:	
What type of data is consent requested for?	*dpv:DataSource, dcat:Resource, smashHitCore:SensorDataCategory, smashitCore:hasContext, smashhitCore:hasMetadata*
What is the source of the data?	*dpv:Processing, smashHitCore:hasMethod, dcat:Resource, dpv:Align, dpv:Erase, dpv:Analyse, dpv:Alter, dpv:Consult, dpv:Record, dpv:Restrict, rdf:hasInputData, rdf:hasOutputData*
What processing will be done to the data?	*dpv:DataSource, smashHirCore:Device, sensor:Sensor, smashHitCore:SensorDataCategory, smashHitCore:hasSensorDataCategory*
Who is responsible for the data processing?	*prov:Agent, prov:Organization, prov:Person, prov:SoftwareAgent, gconsent:hasRole, smashHitCore:DataController, smashHitCore:DataProvider*
Agent questions:	
What entities are involved with the data?	*prov:Agent, prov:Organization, prov:Person, prov:SoftwareAgent, rdf:Role, gconsent:hasRole, dcat:hadRole, smashHitCore:DataController, smashHitCore:agentID, smashHitCore:signatureID*
Who is the data controller?	*rdf:Role, gconsent:hasRole, dcat:hadRole, smashHitCore:DataController, smashHitCore:signatureID*
Who is the data subject?	*rdf:Role, smashHitCore:DataSubject, gconsent:hasRole, dcat:hadRole, smashHitCore:signatureID*
	Continued on next page

Table 2.1 – continued from previous page

Questions	**Relevant smashHitCore concepts and object properties**
How to identify a person?	*dpv:PersonalDataCategory, dpv:Identifying, dpv:Name, dpv:OfficialID, dpv:UID, dpv:Username, dpv:Password, dpv:Contact, smashHitCore:hasContactPoint, dpv:EmailAddress, dcat:hadRole, gconsent:hasRole, smashHitCore:agentID, smashHitCore:signatureID*
How to identify an organisation?	*dpv:PersonalDataCategory, dpv:Identifying, dpv:Name, dpv:Contact, smashHitCore:hasContactPoint, dpv:EmailAddress, dcat:hadRole, gconsent:hasRole, smashHitCore:agentID*
Contract questions:	
What is the type of the contract between the entities?	*fibo-fnd-agr-agr:MutualContractualAgreement, smashHitCore:BusinessToBusiness, smashHitCore:BusinessToConsumer*
Who is involved in the contract?	*prov:Agent, prov:Organization, prov:Person, prov:SoftwareAgent, gconsent:hasRole, dcat:hadRole, dcat:Role, smashHitCore:DataController, smashHitCore:DataSubject, smashHitCore:agentID*
What is the duration of the contract?	*fibo-fnd-agr-smashHitCore:contractID, ctr:hasEffectiveDate, smashHitCore:hasCreationDate, smashHitCore:hasEndDate, smashHitCore:hasExpirationDate, smashHitCore:hasMinimumDuration*
What does a contract include?	*fibo-fnd-agr-smashHitCore:contractID, ctr:hasContractualElement, smashHitCore:TermsAndConditions, odrl:Policy, odrl:Duty, smashHitCore:Obligations, smashHitCore:obligationID, smashHitCore:hasSignatures, fibo-fnd-agr-ctr:hasBeneficiary*
What is the status of a contract?	*smashHitCore:contractID, smashHitCore:ContractStatus, smashHitCore:hasContractStatus, smashHitCore:Created, smashHitCore:Updated, smashHitCore:Expired, smashHitCore:Pending, smashHitCore:Renewed, smashHitCore:Signed, smashHitCore:Terminated*
What are the states of the contract obligations?	*smashHitCore:contractID, smashHitCore:Obligations, smashHitCore:obligationID, smashHitCore:ObligationState, smashHitCore:hasPendingState, smashHitCore:hasFulfilled, smashHitCore:hasInvalid, smashHitCore:hasValid*

2.5 The smashHitCore Ontology

This section presents an overview of smashHitCore's main concepts (i.e., consent, contracts and licenses, information entity, agent, role, data source, resource, data processing, location and time) and how they relate to each other. smashHitCore consists of 202 classes, 87 object properties, 42 data properties and reuses 9 ontologies: GConsent (Pandit et al., 2019a), DPV (Pandit et al., 2019b), FIBO (EDM Council, 2020), PROV-O (Lebo et al., 2013), OntoSensor (Russomanno et al., 2005), schema.org (W3C Community Groups, 2012), Data Catalog Vocabulary (DCAT) (World Wide Web Consortium and others, 2014), CampaNeo [3] and Languages, Countries, and Codes (LCC)(Object Management Group, 2019). The description of the ontology follows the guidelines for minimum information for the reporting of an ontology (MIRO) (Matentzoglu et al., 2018). The full specification of smashHitCore is available online (The smashHit project, 2022).

2.5.1 Consent

To model the concept of consent in a GDPR-compliant manner based on Art 4(11), smashHitCore reuses the class *gconsent:Consent* from GConsent (Pandit et al., 2019a). Consent is a dynamic entity (i.e., its status and purpose can change over time) that can have status (*smashHitCore:ConsentStatus*) such as *gconsent:Invalid, gconsent:Withdrawn, gconsent:Valid, smashHitCore:Granted.* Further, consent has provenance information such as *smashHitCore:GrantedAtTime, smashHitCore:WithdrawnAtTime, smashHitCore:atLocation* that is associated with it.

To model detailed information about the context in which a consent decision is made, we have reused the *gconsent:Medium* class, which describes the medium through which consent was provided (e.g., web form, app, paper). Consent can be associated with a specific purpose via the *dpv:hasPurpose* object property with range the class *dpv:Purpose* (e.g., *dpv:CommercialInterest, dpv:Security* etc.). Consent can be related to specific data source and processing via the properties *smashHitCore:isAboutData* and *smashHitCore:forDataProcessing.* Any type of processing *smashHitCore:hasInput* and *smashHitCore:hasOutput* (e.g., sensor data from a specific sensor such as *sensor:GPS*). An overview of the class consent (in white) and relationships to other classes (in blue) are presented in Figure 2.1.

[3]https://github.com/STIInnsbruck/CampaNeoUI/blob/master/ontology/CampaNeo.owl

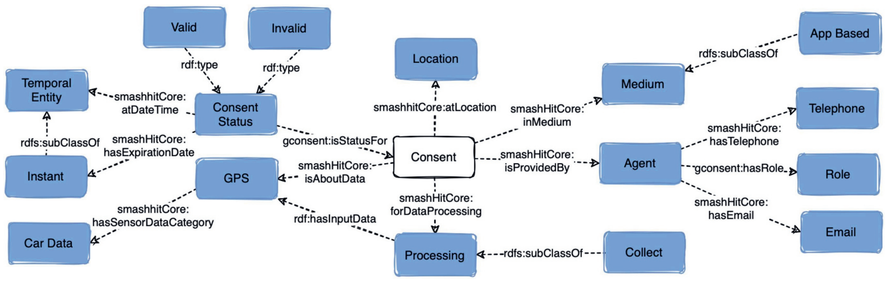


Figure 2.1: Overview of the class *Consent* and the classes related to it in smashHitCore.

2.5.2 Contracts

smashHitCore reuses *fibo-fnd-agr-ctr:Contract* (Figure 2.2) to represent contracts and contractual obligations related to data sharing for UC1 and UC2. A challenge here is to provide all the necessary information needed for a contract while having a generic enough semantic model that can be utilised in diverse contract-based scenarios. The class *fibo-fnd-agr-ctr:MutualContractualAgreement* from FIBO (EDM Council, 2020), which can model contracts for both UC1 and UC2 was reused in smashHitCore. We have represented two widely used types of contracts, namely *smashHitCore:BusinessToBusiness* and *smashHitCore:BusinessToConsumer*. The object property *smashHitCore:hasDataProcessor* links a contract to a specific contract agent such as *smashHitCore:DataProcessor*.

Contracts are comprised of various building blocks such as terms and conditions, which can be linked to a contract via the object property *samshHitCore:hasTermsAndConditions*. When a contract has status *smashHitCore:Created*, the obligations associated with it become active and (i.e., all contractors need to start adhering to them). If a contract has expired, then all obligations become invalid. To capture this information, we have modelled different obligation states with the class *smashHitCore:ObligationState*, namely *samshHitCore:Invalid*, *samshHitCore:Valid*, *samshHitCore:Pending*, *smashHitCore:Fulfilled*.

Metadata such as the contract's creation date and its effective date can be recorded as well with the object properties *smashHitCore:hasCreationDate* and *fibo-fnd-agr-ctr:hasEffectiveDate*. The object property *smashHitCore:hasExpirationDate* has been defined to represent the expiration date of a contract, while *smashHitCore:hasEndDate* refers to the date when a contract is terminated ahead of its expiry date. To ensure the integrity of contracts and to allow contract and identity verification, the data property *samshHitCore:hasSignature* can be used to store contractors' signatures (in xds:string format).

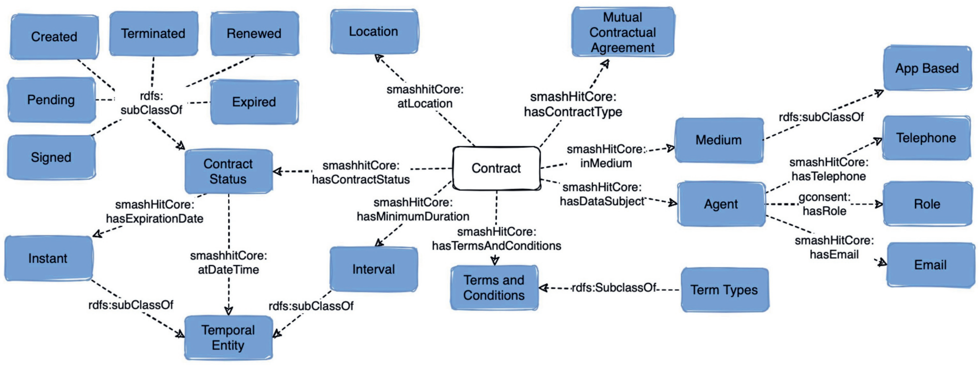


Figure 2.2: Overview of the class *Contract* and the classes related to it in smashHitCore.

2.5.3 Information Entity

The class *smashHitCore:Information Entity* (Figure 2.3) models concepts that are used to provide specific details about consent and contracts. The subclass *smashHitCore:SensorDataCategory* describes several categories of sensor data needed for our UC1, specifically *smashHitCore:BuildingData, smashHitCore:CarData* and *smashHitCore:RoadData.*

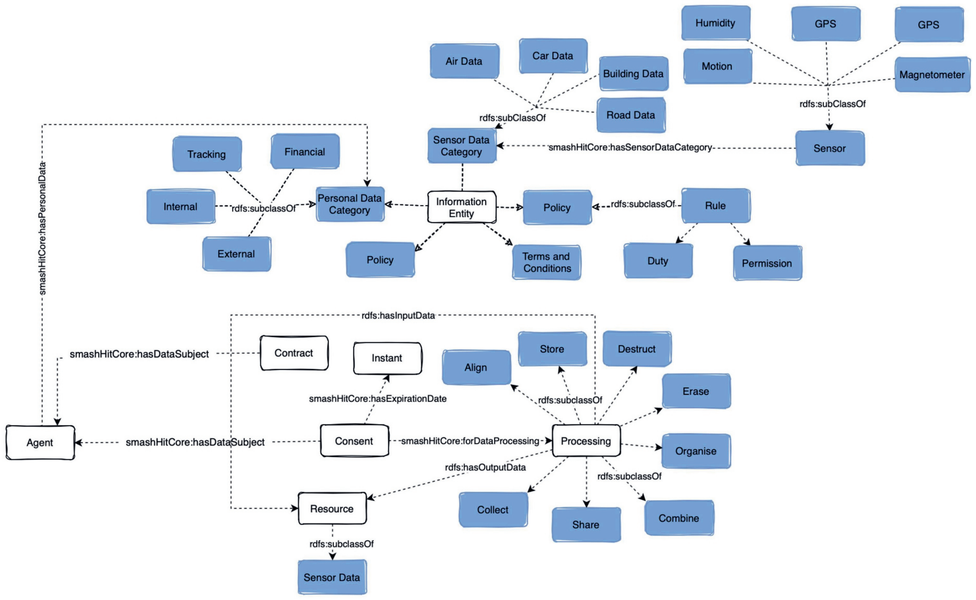


Figure 2.3: Class *Information Entity* and its connection to other concepts.

Four types of personal data (*dpv:PersonalDataCategory*) have been modelled based on GDPR's Art. 4.1 (i.e., *dpv:Internal*, *dpv:External* and *dpv:Tracking*). These types of data are needed for processes such as user verification within the smashHit project, for insurance purposes based on UC2 and for modelling contracts. The terms and conditions (*smashHitCore:TermsAndConditions*) as well as the policy (*odrl:Policy*) are an essential part of any contract and license. The terms and conditions are a set of special conditions for a contract, while a policy defines a set of prohibitions and permissions (Iannella, 2004).

2.5.4 Agent and Role

The class *prov:Agent* (Figure 2.4) represents three types of entities (*prov:SoftwareAgent, prov:Person, prov:Organization*) that are currently present in UC1 and UC2. Several types of organisations were reused from the FIBO (EDM Council, 2020) and CampaNeo [4] ontologies (*fibo-fbc-fe-fse:InsuranceCompany, campaneo:AutomobileOrganisation*, etc.). When designing the ontology we also considered that the role (class *dcat:Role*) of an agent can change over time and that an agent can have multiple roles depending on a given context and as defined by GDPR (*smashHitCore:DataController, smashHitCore:DataSubject, smashHitCore:DataProcessor, smashHitCore:DataProtectionOfficer*). An agent can be linked to a current or past role by using the *gconsent:hasRole* and *dcat:hadRole* object properties. Further, agents have personal data associated with them, which is represented by the class *dpv:PersonalDataCategory*. In cases such as GDPR compliance verification when all entities need to be notified if non-compliance has been detected, smashHitCore models a contact point (*dpv:Contact*) of an agent with the classes *dpv:Email* and *dpv:TelephoneNumber*.

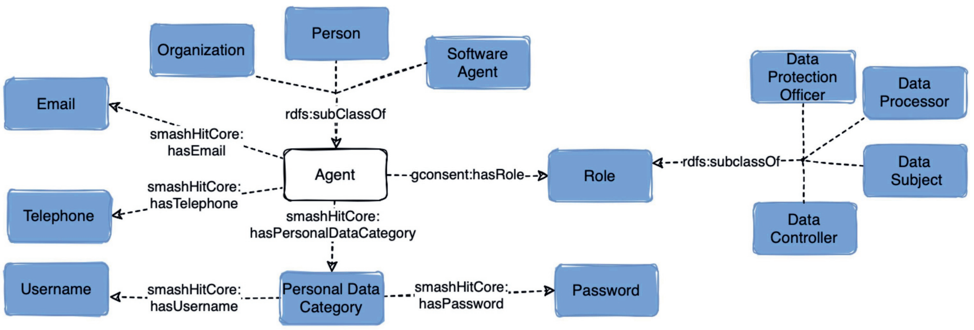


Figure 2.4: Overview of class *Agent* and the classes related to it in smashHitCore.

2.5.5 Data Source and Resource

The class *dpv:DataSource* models the source of the data for which consent, contract or license is needed. The source can be a device such as a sensor (see Figure 2.3) (the

[4]https://github.com/STIInnsbruck/CampaNeoUI/blob/master/ontology/CampaNeo.owl

class *sensor:Sensor* is reused from OntoSensor (Russomanno et al., 2005)), which models a wide spectrum of sensor concepts relevant to our UC1 and UC2. GDPR requires consent to be specific and unambiguous thus we have also reused specific types of sensors such as *sensor:GPS*, *sensor:Motion*. We have also defined complex sensors such as *smashHitCore:PhotoelectricSensor, smashHitCore:Magnetometer, smashHitCore:AirPollutionSensor*. Further, the concept of a resource (*dcat:Resource*) is reused from DCAT (World Wide Web Consortium and others, 2014). A resource is an asset such as a *dcat:Dataset, smashHitCore:PersonalData* or *smashHitCore:SensorData*, which has been produced by an agent. For example, in UC1 and UC2, a data set generated with vehicle sensor data can be both the input and the output of data processing.

2.5.6 Data Processing

Consent must be given for a specific type of data processing. smashHitCore reuses the concept of data processing from DPV (Pandit et al., 2019b) and relevant concepts that describe different types of data processing such as *dpv:Adapt*, *dpv:Align*, *dpv:Collect*, *dpv:Record* and *dpv:Use*. We have limited the concepts to those events directly listed within GDPR (Art. 4 (2)) and hence the result is similar to the processing concepts from GConsent (Pandit et al., 2019a). Each data processing event has input and output, which we represent via the *rdf:hasInputData* and *rdf:hasOutputData* properties.

2.5.7 Location and Time

To represent the location and time of consent and contract status updates (e.g., when and where consent was granted or a contract was signed), smashHitCore reuses the concepts *LCC:Location* and *time:TemporalEntity*. The *LCC:Location* class has the subclass *LCC:GeographicRegion*, which is used to describe the geographic location where the status of consent or a contract changes. Its subclass *smashHitCore:StorageLocation* represents the technical storage location defined by a URL (e.g., a URL of a server where the data is stored).

The class *time:TemporalEntity*, which is reused from OWL-Time (Cox, 2022), is a temporal sequence that represents a time instant (e.g., the expiration date of consent or a contract) or a period of time (e.g., duration of data progressing, consent or a contract). *time:TemporalEntity* has two subclasses *time:Interval* and *time:Instant*. The object properties *time:hasBeginning* and *time:hasEnd* can be used to express the start and end of a time instant.

2.6 Application of smashHitCore for GDPR Compliance

smashHitCore, which is a basis for the smashHit legal knowledge graph (KG) (see Figure 5b in (Chhetri et al., 2022b)), has been used in the compliance verification tool (Chhetri et al., 2022b; Tauqeer et al., 2022) that is developed as a part of the smashHit project to enable legally compliant data exchange in the smart city and insurance use case scenarios (see Section 2.2). Our work (Chhetri et al., 2022b; Tauqeer et al., 2022), which makes use of the smashHitCore ontology, has been evaluated in real-world insurance and smart city scenarios. Details regarding the integration and testing are available in ("smashHit Demonstrator Videos", 2022; "smashHit Whitepapers", 2022).

The semantic annotation of one's informed consent and contracts (i.e., building the KG), their management and use for compliance are the main tasks of our automated compliance verification tools presented in (Chhetri et al., 2022b; Tauqeer et al., 2022). In (Chhetri et al., 2022b; Tauqeer et al., 2022), by following a microservices (Dragoni et al., 2017) architecture, which supports scalability (see (De Lauretis, 2019)), we have presented two tools that fully automate the GDPR compliance verification processes for both consent and contract. The tools adhere to the principle of data protection by design (Rec. 78) and are built around UC1 and UC2 modelled by the smashHitCore ontology. Moreover, Knoke and Iheanyi (Knoke & Nwankwo, 2022) provide details on the technical and organisational measures that our work implements for data protection by design principles from the maturity model perspective.

The consent compliance verification tool (Chhetri et al., 2022b), which makes use of the smashHitCore ontology, is comprised of several components, each having a different task. Data and consent management are performed by the data processing module, which also deals with the execution of different SPARQL (SPARQL Working Group, 2008) queries (i.e., consent annotation, consent revocation) based on the smashHitCore ontology. The consent module in the tool performs the validation of the input consent, which is received in JavaScript Object Notation (JSON) and also transforms it to the KG following the smashHitCore ontology. A hybrid algorithm, combining both asymmetric and symmetric encryption, is used to ensure the integrity of the information and to make the KG access secure. Due to the sensitive information stored in the KG, it is not openly available. Its use within smashHit is perfomed via secured APIs (see Figure 6 in (Chhetri et al., 2022b)). Details on the automated compliance verification tool, its mechanism and how compliance is performed with the help of smashHitCore are presented in (Chhetri et al., 2022b). In addition, smashHitCore is used as the main schema for building GDPR-compliant consent request forms online as showcased in (Rasmusen et al., 2022) and for visualising post-consent data flows with KGs in (Bless et al., 2021).

2.7 Evaluation

In this section, we present the evaluation of smashHitCore in terms of ontology engineering, expressivity (ability to represent diverse GDPR and sensor data concepts) and its application for automated consent and contract compliance. Section 6.4 provides information on smashHitCore's evaluation with a set of competency questions and standard ontology evaluation tools, while Section 2.7.2 provides details on the performance analysis of the tools for consent and contract verification that utilise smashHitCore.

2.7.1 Ontology Evaluation

To evaluate smashHitCore's granularity and completeness with regards to UC1 and UC2, a set of competency questions (see Table 2.1) for consent, contracts, data processing, and corresponding agents was used. The competency questions for consent presented in Table 2.1 were derived from GDPR's requirements for informed consent (Art. 7, Rec. 32, 42, 43) and are similar to the ones used in (Pandit et al., 2019a). We have also derived similar competency questions for contracts which are based on GDPR's requirements for data sharing contracts (Art. 25 and 32). For each competency question we provide the set of relevant concepts and object properties that can be used to model the answer. Similarly to (Pandit et al., 2019a), smashHitCore was also evaluated with the HermiT (Shearer et al., 2008) and Pellet (Sirin et al., 2007) reasoners, and no inconsistencies were found. The OOPS! (Poveda-Villalón et al., 2014) ontology pitfall scanner was used as well to detect and correct issues in our semantic model.

The evaluation with the competency question in Table 2.1 shows that our ontology can model informed consent based on GDPR's requirements and can further provide details about specific types of sensors and sensor data, which is not possible when using ontologies such as GConsent (Pandit et al., 2019a), CDMM (Fatema et al., 2017) in a stand-alone way. smashHitCore represents different types of agents, contact information and specific types of personal data (e.g., email, username, address), which are not represented in detail in ontologies such as CDMM, GConsent, SPL and SPLog (Kirrane et al., 2020). Such information can be useful for processes such as consent verification and compliance checking as presented in (Chhetri, 2021; The smashHit project, 2021). Further, smashHitCore also models contracts, which allows it to be reused in scenarios where consent is not enough for the legal processing of data. In such cases, existing contract ontologies such as FIBO (EDM Council, 2020) can be reused but they rarely model consent. smashHitCore provides a semantic representation of these concepts in one place.

2.7.2 Interoperability and Performance Evaluation

The two main tools presented in (Chhetri et al., 2022b; Tauqeer et al., 2022), which utilise the smashHitCore ontology, were developed as a part of the smashHit project. These tools were further used by other use case-specific software components that were also part of the smashHit project, namely data use traceability (smashHit consortium, 2022a) and the context sensitivity solution (smashHit consortium, 2022c) in the connected car and smart city feedback application ("smashHit Demonstrator Videos", 2022). Therefore, we evaluate the interoperability and performance based on the application of the smashHitCore ontology.

Interoperability. The interoperability of the smashHitCore ontology is evaluated using three dimensions: (i) semantic interoperability, (ii) technical interoperability, and (iii) organisational interoperability, following Ducq and Chen (Ducq & Chen, 2008) and Guédria et al. (Guédria et al., 2015). Semantic interoperability enables a system to combine information from heterogeneous sources and process it in a meaningful manner, and organisational interoperability focuses on business goals, i.e., use cases in our case. Similarly, organisational interoperability is concerned with interconnecting different services including data integration and exchange (Guédria et al., 2015). The smashHitCore ontology models all of the necessary information needed for representing consent and contract, that is required by the use cases in order to share data and design applications, such as user interfaces, in a standardised manner (smashHit consortium, 2022b), thereby achieving organisational interoperability. As with organisational interoperability, the smashHitCore ontology also meets the requirement of semantic interoperability, as it enables the exchange of the same meaning of information across different systems in a machine-readable format. For instance, all software components, such as data use traceability (smashHit consortium, 2022a), can interpret the same meaning as (Chhetri et al., 2022b) for consent-related information, meeting the requirement of semantic interoperability. The third dimension is technical interoperability. For technical interoperability, we check to see if the ontology in any way supports technical interoperability, enabling data exchange and the integration of various software components. In our case, the smashHitCore ontology has supported the development of an application based on a common specification and has enabled the seamless integration of multiple software components, including those from use cases (smashHit consortium, 2022a, 2022b; "smashHit Demonstrator Videos", 2022). We can therefore conclude that the smashHitCore ontology facilitates interoperability. Similar to interoperability, the use of the smashHitCore ontology in the applications did not impede scalability and had an acceptable performance for our use cases (Chhetri et al., 2022b; Tauqeer et al., 2022).

Performance. The performance of the smashHitCore ontology is evaluated by measuring the execution/processing time of queries to/from GraphDB (Sirma Group, 2019). This is performed for both of the applications of smashHitCore (see Section 2.6) based on the use cases discussed in Section 2.2. As an example, in the Contract Compliance Verification (CCV) (Tauqeer et al., 2022) tool, a contract creation process (i.e., annotating standard contract information such as terms and conditions, clauses, contracting party information, and party signature) took an average of 777.1 milliseconds with

2938 bytes of data to create a new contract based on 10 experiments. The GDPR contract compliance verification took an average of 516.6 milliseconds based on 5 experiments. Similarly, in (Chhetri et al., 2022b), a consent creation process took 7.3 s while compliance checking based on consent took 6.6 s. One of the main factors that affected the performance times is the complexity of contracts, which contain noticeably more information than consent.

2.8 Conclusions

In this paper, we presented the smashHitCore ontology that goes beyond existing work by representing both consent and contracts as legal GDPR bases for sensor data processing and sharing. The ontology is openly available and currently used as a schema for the smashHit KG (details in (Kurteva, 2023)). The utilisation of smashHitCore for consent and contract visualisation (see (Bless et al., 2021; Rasmusen et al., 2022; Rasmusen, 2022)) and for automated GDPR compliance (see (Chhetri et al., 2022b; Kurteva, 2021)) has proven the ontology's ability to successfully link two complex GDPR legal bases (consent and contracts) for our UC1 and UC2 in a meaningful way. The evaluation results have also shown that GDPR compliance verification can be performed in a reasonable amount of time with smashHitCore as the main underlying data model.

The following limitations of smashHitCore can be noted. Our ontology focuses primarily on GDPR-compliant consent- and contract-based vehicle sensor data sharing and processing in two use cases, while GDPR is the leading law on EU citizens' data protection, each EU member state has its own country-specific legislations, procedures and data protection authorities. Finally, we have focused our work primarily on web-based (digital) consent and contracts thus processes such as their digitization have not been semantically modelled.

Ontologies as a type of digital asset require maintenance and updates over time. Future work might include expanding the scope of smashHitCore based on, but not only, its continuous use in UC1 and UC2. For example, extending the class *dpv:Purpose*, in smashHitCore, with different medical purposes and treatments from the Informed Consent Ontology (ICO) (Lin et al., 2014), which may or may not be covered by one's insurance policy. Currently, smashHitCore reuses the class *dpv:Risk*. We plan to define specific risks when it comes to (personal) sensor data sharing in UC1 and UC2 as well. Extending the smashHitCore ontology with the DALICC (Pellegrini et al., 2019) semantic model for digital licensing is also planned as future work. Licensing with DALICC can reduce transaction costs (Pellegrini et al., 2019) and can improve data sharing of digital assets (e.g., contracts, personal datasets). An idea to be explored is the use of an upper ontology such as the Basic Formal Ontology (BFO) (Otte et al., 2022) for alignment of our ontology with others to support smashHitCore's wider reuse. This can be conducted by aligning different processes that smashHitCore models (e.g., data processing, data sharing, giving and revoking consent, signing contracts, compliance verification) with BFO's *Process* class at different levels. For instance, GDPR compliance verification is a complex process that can be divided into sub-processes such as identity verification, informed consent request, data use traceability etc. Each sub-process has temporal characteristics that are essential for performing adequate GDPR compliance verification. Although all this information can already be captured with smashHitCore, BFO's classes *Process* and *ProcessProfile* can enhance the granularity of process details and simplify the reuse of smashHitCore beyond its current scope (BFO version 2.0 specification in (BFO Discussion Group, 2007)).

Author Contributions: A.K.: Conceptualisation, methodology, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing, visualisation; T.R.C.: Conceptualisation, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing; R.H.: Validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing; A.T.: Conceptualisation, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing; A.F.: Supervision, investigation, writing—review and editing, funding acquisition; K.N.: Conceptualisation, methodology, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing; A.C.: Conceptualisation, methodology, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing; A.Z.: Conceptualisation, methodology, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing; S.S.: Conceptualisation, investigation, resources, writing—original draft preparation, writing—review and editing; T.F.: Conceptualisation, investigation, resources, writing—original draft preparation, writing—review and editing; E.D.: Supervision, investigation, writing—review and editing, funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work is part of the smashHit H2020 project (grant agreement No 871477).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The smashHitCore ontology is available at https://smashhiteu.github.io.

Acknowledgments: This work is part of the smashHit H2020 project (grant agreement No 871477). We thank Volkswagen AG, LexisNexis Risk Solutions, Forum Virium Helsinki, Infotripla and ATOS for providing the use cases and supporting our collaboration. Further, we thank the legal team from University of Hannover (LUH/IRI) for their legal advice during the ontology specification.

Conflicts of Interest: The authors declare no conflict of interest.

CHAPTER 3

Automated GDPR Contract Compliance Verification Using Knowledge Graphs

WAGENINGEN
UNIVERSITY & RESEARCH

This chapter is published as:

Tauqeer A., Kurteva A., Chhetri TR., Ahmeti A., Fensel A. Automated GDPR Contract Compliance Verification Using Knowledge Graphs. Information. 2022; 13(10):447. https://doi.org/10.3390/info13100447 (Tauqeer et al., 2022)

Abstract

In the past few years, the main research efforts regarding General Data Protection Regulation (GDPR)-compliant data sharing have been focused primarily on informed consent (one of the six GDPR lawful bases for data processing). In cases such as Business-to-Business (B2B) and Business-to-Consumer (B2C) data sharing, when consent might not be enough, many small and medium enterprises (SMEs) still depend on contracts—a GDPR basis that is often overlooked due to its complexity. The contract's lifecycle comprises many stages (e.g., drafting, negotiation, and signing) that must be executed in compliance with GDPR. Despite the active research efforts on contracts, contract-based GDPR compliance and challenges such as contract interoperability have not been sufficiently elaborated on yet. Since knowledge graphs and ontologies provide interoperability and support knowledge discovery, we propose and develop a knowledge graph-based tool for GDPR contract compliance verification (CCV). It binds GDPR's legal basis to data sharing contracts. In addition, we conducted a performance evaluation in terms of execution time and test cases to validate CCV's correctness in determining the overhead and applicability of the proposed tool in smart city and insurance application scenarios. The evaluation results and the correctness of the CCV tool demonstrate the tool's practicability for deployment in the real world with minimum overhead.
Keywords— contracts, data sharing, ontology, Knowledge graph, GDPR compliance, smart cities, insurance.

3.1 Introduction

The General Data Protection Regulation (GDPR) ("GDPR", 2018), which came into effect on 25 May 2018 across all European Union (EU) member states, lays down strict requirements for the processing, storing, and management of EU citizens' data (Li & Samavi, 2018; "European Parliament and Council", 2016). The following six legal bases are defined in GDPR Art. 6 that justify the processing of personal data ("European Parliament and Council", 2016): (i) informed consent; (ii) performance of a contract; (iii) legal obligation; (iv) protection of vital interests of the data subject; (v) performance of tasks carried out in the public interest or in the exercise of official authority vested in the controller; and (vi) legitimate interest pursued by a data controller. At the minimum, one of these six bases must be met for the lawful processing of personal data, which is defined as *"any information relating to an identified or identifiable natural person"* (Art. 4 (1), 5 and 6).

In most data sharing scenarios, organisations focus on collecting informed consent from the data subject (i.e., an identifiable natural person) (Art. 4 (1)). For instance, collecting data about an individual's online browsing behaviour is based on consent, which can be collected via cookie banners (Habib et al., 2022). However, consent is not always enough, for example, in the case of online services where a contract is required. In scenarios such as online services (e.g., information society services), the European Data Protection Board (EDPB) (European Data Protection Board, 2022) also highlighted the necessity of a contract by issuing new guidelines in 2019 ("Contractual necessity", 2019).

There are other scenarios, such as Business-to-Business (B2B) and Business-to-Consumer (B2C) data sharing, in which consent is not sufficient. This is due to the need for specific terms and conditions and their complexity. These terms and conditions yield specifications for an agreement between all contractual parties regarding what is allowed and not allowed. In addition, some of the main differences between consent and contracts are the following: (i) consent can be revoked at any time, while a contract cannot be terminated before its minimum duration; (ii) consent has predefined clauses, while contract clauses can be negotiated until an agreement is reached by all involved parties.

There is an opportunity for organisations to manage personal data under the GDPR in digital contracting (i.e., a process that transforms the entire contract lifecycle into digitalised and collaborative workflow). However, many organisations in insurance, mobility, and smart cities domains still face challenges in binding GDPR rights with businesses and fail to comply with GDPR, specifically when contracts are used (Li et al., 2019). Another persistent challenge for many small and medium enterprises (SMEs), especially in the mobility and insurance sectors within the smart cities' domain, is the unprecedented amount of data. These data are generated every day and spread across different silos (organisations, departments, people, and databases) (Kurteva et al., 2021). In our scenarios, data include contractual information, which can number in thousands per month in an organisation such as LexisNexis Risk Solutions, a leading insurance data provider. Locating specific data and permissions for its sharing, such as consent and contracts, can be time-consuming and computationally expensive due to the lack of interoperability. We can define the primary challenges regarding contracts and their management as (i) building interoperable GDPR-based contract models and (ii) performing GDPR contract compliance verification (CCV). *In digital contracting or agreements, the CCV is a process that ensures data processing according to GDPR by detecting contractual conflicts or breaches. A contract breach or conflict is a failure without legal excuses to perform any promise that forms all or part of the contract* ("Breach of Contract", 2022); (iii) monitoring contract execution; and (iv) updating contracts and the involved contractual parties accordingly.

While challenges vary in complexity, all should be solved in a scalable and secure GDPR-compliant manner. To do so, organisations have to adopt security measures and data protection on contracting services and processes regarding GDPR (Art. 25, 32). According to Art. 28 (3), contracts must include the following details: (i) the subject matter and duration of the processing; (ii) the nature and purpose of the processing; (iii) the type of personal data and categories of the data subject; and (iv) controller's obligations and rights. It also sets the minimum required contractual clauses, such as the duty of confidence, security measures, data subject rights, audits, and inspections. Data subject rights protection is another crucial challenge for organisations according to GDPR (Art. 25). Organisations must implement GDPR technical organisational measures (TOMs). GDPR TOMs comprise all provisions put into place to guarantee the security of personal data, such as pseudonymisation. These provisions implement data-protection principles such as data minimisation (see details in Section 3.4.2.1).

This paper focuses on two scenarios from the smashHit (The smashHit project, 2020a) project, which aims to develop a scalable, trusted, and secure solution for data sharing and contract management in the connected car and smart cities domains. Use case 1 (UC1) focuses on data sharing in the insurance domain, where data sharing is key to informed decision-making. Although informed consent is the main legal basis for data sharing between the data subject and the data processor (i.e., an insurance company), when data are sold or analysed by third-party entities, additional terms and conditions must be presented and agreed upon. These form the basis of a contract, which becomes the main legal basis. Use case 2 (UC2), on the other hand, presents a data sharing scenario in the smart cities domain, where an unprecedented amount of data (contractual information) is simultaneously emitted, shared, and analysed by multiple agents (i.e., software, humans, and organisations). In cases such as B2B data sharing in UC2, a contract is also needed as it provides in-depth specifications of each contract party's obligations and the specific terms and conditions that need to be followed.

Research on GDPR compliance for contracts has started to gain popularity, especially in the Semantic Web domain. Solutions such as smart contracts (based on

Blockchain (Hunhevicz et al., 2022; Liu et al., 2022)) and semantic contract or agreement (Fensel et al., 2020; Hogan et al., 2021; Pandit et al., 2019c; “Semantic Agreement”, 2022) based on knowledge graphs (KGs) are commonly used for digital contracting. However, GDPR rights, such as the exercise of rights to the eraser and right to rectification, identifying the data controller or data processor, and data transfer are still persistent issues in smart contracts (Jusic, 2020; Voss, 2021). A further challenge with smart contracts is the classification of the various contractual parties (e.g., joint controllers) involved. The possible misclassification can directly affect the contractual party’s responsibilities under the law and their potential liability for violations. These contracts define rules and penalties for an agreement and automatically enforce those contractual clauses. In digital contracting, machines do not always understand contractual terms. In such a case, smart contracts cannot handle these contractual terms that are vague (“Smart Contract”, 2020). GDPR compliance using blockchain technology for data processing results in compliance issues due to the different methods used to ensure privacy-by-design and privacy-by-default (Jusic, 2022).

KGs and ontologies can aid the building of common solutions, foster interoperability, support knowledge discovery, and decision making (Breitfuss et al., 2021; Chhetri et al., 2022a; Fensel et al., 2020; Hogan et al., 2021; Pandit et al., 2019c). In UC1 and UC2 as discussed above, the usage of web technologies in combination with semantic technologies ensures information reusability, reliability, and inference to support the end users on the web (Sermet & Demir, 2021). The use of KGs for consent-based GDPR compliance has already proven to be beneficial in our previous work presented in (Chhetri et al., 2022b). In our earlier work (Chhetri et al., 2022b), we proposed and developed a scalable data protection by design tools for automated compliance verification and auditability based on informed consent using KGs. This research focuses on performing GDPR contract compliance verification, where consent is not enough, for example, in online services (“Contractual necessity”, 2019). Furthermore, in comparison to the diverse consent ontologies that are available, as shown in (Kurteva et al., 2021), there are few ontologies that model contracts based on GDPR. Following this and our previous work in (Chhetri et al., 2022b), we present a KG-based solution for digital contracting, which has the following functionalities: (i) binding GDPR with data sharing contracts and (ii) performing CCV checks on contracts.

The main contributions of our work are as follows:

1. A scalable tool for managing semantic-based contracts within smart cities and insurance use cases;
2. Our tool implements a KG-based approach for GDPR-compliant CCV;
3. An ontology and KG for contracts that can be reused in various cases and domains.

We would like to emphasise that our tool reduces contractual execution time and cost compared to manual contract compliance verification. With the example of the contract repository, contractors can easily track their data usage and obtain compliance notifications within the tool. Last but not least, our tool improves contract management processes, which ultimately reduces the overall contracting cost compared to the (classical) manual contracting approach or to ad hoc solutions that each come with their vendor lock-in solutions.

The rest of the paper is structured as follows. Related research studies are presented in Section 4.2. We describe the approach for building this tool in Section 6.3. The tool's architectural design and implementation are presented in Section 3.4. We discuss the evaluation and results in Section 6.4. Finally, the conclusion and future research are presented in Section 4.7.

3.2 Related Work

This section presents an overview of related work on contract management (Section 3.2.1), semantic contract modelling (Section 3.2.2), and on contract-focused GDPR compliance verification (Section 3.2.3).

3.2.1 Contract Management

Longo et al. (Longo et al., 2015) present a model for the construction and management of Service Level Agreements (SLAs) by extending the XML-based WSLA (Service Level Agreements for web services) framework (Keller & Ludwig, 2003). Two of the main challenges that were solved include (i) the lack of standard models representing service contracts and their SLAs in service-oriented architecture (SOA) and service network environments and (ii) making SLAs effectively machine-readable.

The first one is solved by complementing WSLA with composition topologies and rules. They achieve the second one by modelling the template as a digraph that is implemented in a NoSQL (Not only Structured Query Language) [1] graph DBMS (Database Management Systems). The evaluation of the functionalities was performed based on the following five metrics; (i) availability; (ii) response time; (iii) mean time to repair; (iv) mean time to failure; and (v) mean time between failure. Based on these assessment capabilities, they offer a tool named DAMASCO (Data Manager for Service Composition) to Information Technology (IT) professionals during the design phase. However, this tool does not comply with GDPR for data processing.

Guo et al. (Guo et al., 2021) present an electronic contract management system based on blockchain technology for commodity procurement in the electric power industry. The proposed BEcontractor process-oriented contract management system solves a series of security issues (e.g., signatures and digital certificates) existing in traditional

[1] https://www.ontotext.com/knowledgehub/fundamentals/nosql-graph-database/

contract management systems. The evaluation of the system has shown that it can significantly reduce the time and cost of completion of the contract signing process. With this, they also present that the payment period is shortened from three months to around one month. BEcontractor works under China's legal protection of electronic contracts, but there is no information about data processing rights, such as GDPR, and there is no information about B2B contracts.

Voronova (Voronova, 2020) proposes a contract management system, which provides a classification of contracts (e.g., sales contract and supply contract) and their types (e.g., unilateral agreement and bilateral treaty) for network trading companies. The contract types are based on several features, such as the rights and obligations of interested parties to the contract. The author emphasises determining the contract strategy by choosing the contractual structure (based on types, sequence of conclusion, and relationship of contracts), setting the key performance indicator (KPI) of the contract, the KPI of business processes, and establishing relationships between them. The author provides guidelines for the organisations to improve their efficiency and competitiveness and to protect their interests by improving the efficiency of the contract's management. However, there is no information about data processing under GDPR.

Schmidt et al. (Schmidt et al., 2019) propose an electronic Contract Management System (eCMS) in the health domain. The primary objective of eCMS focuses on Continuous Process Improvement (CPI) in eCMS to align best with Lean Six Sigma and Quality Management frameworks. The authors standardised the processes and increased both the system's productivity through workflow design and its efficiency and achieved improved quality with respect to the eCMS process. The Cobblestone for eCMS web-based system, which provides contract tracking, drafting, and administration functionalities, was selected. It also offers contract lifecycle management that streamlines and automates the entire contract process from contract drafting to completion. However, there is no information about how personal data are treated in the eCMS and what types of contracts are available.

Simić et al. (Simić et al., 2021) explored the applications of smart contracts in the legal domain and proposed a blockchain-based smart contract management system with a user interface for end-user accessibility. The authors conclude that without any intermediary involvement, smart contracts can be concluded more efficiently and can reduce the contract's cost (Simić et al., 2021). From the underlying mechanisms of the blockchain, there are many advantages, such as no risk of data loss and malicious data manipulation arising. For potential disputes, smart contracts should provide a mechanism to resolve them fairly (Simić et al., 2021). For these potential disputes, the contractual parties have to rely on the legal system. Despite the system being open-access, GDPR's legal basis (necessary for lawful data processing in contracts) has not been discussed.

3.2.2 Semantic Modelling

Zou et al. (Zou et al., 2010) present a formal service contract model for cloud service and accountable Software as a Service (SaaS) by utilising semantic technologies. The model allows service providers and consumers to monitor the execution of service contracts and to keep track of obligation fulfilment during service delivery. They propose a graphical model based on Colored Petri-Nets (CPN) to model contract obligations and their interdependencies. However, this service contract implementation supports only B2C contracts and does not comply with GDPR because it was developed and implemented before GDPR enforcement.

Perrin and Godart (Perrin & Godart, 2004) propose a semantic-based contract model to describe business interactions, deploying cross-organisational activities (called synchronisation points) and enforcing and controlling policies. A rule-based approach is used for this model. The resulting contract model describes the processing of web services for cooperation and the enforcement of contract clauses by synchronisation points. Similarly to (Zou et al., 2010), the work in (Perrin & Godart, 2004) focuses only on B2B contracts and was conducted before the acceptance of the legislation; thus, its compliance is questionable.

Kabilan and Johannesson (Kabilan & Johannesson, 2003) present the Multi-Tier Contract Ontology (MTCO), which consists of three layers. The first layer defines conceptual models of contracts, while the second layer is responsible for defining specific types of contracts. The third layer defines contractual obligation and their fulfilment patterns. Furthermore, MTCO models have different stages with respect to the contract-signing process (e.g., conception, drafting, and signing), which can be beneficial for modelling contracts in detail (e.g., to provide provenance information). In addition, MTCO models contract details such as performance obligations, rules, rights, and payments. However, the ontology does not clearly differentiate between traditional contracts and electronic contracts.

Cesare and Geerts (de Cesare & Geerts, 2012) present an ontology for contracts, which consists of following three building blocks: (i) agreements amongst persons, (ii) promises, and (iii) considerations. The ontology in (de Cesare & Geerts, 2012) model contracts include types (e.g., verbal and written), events related to the execution, fulfilment, and the exchange of contracts. However, modelling specific contract domains (e.g., in sales) and the formalisation of the ontology in Web Ontology Language (OWL) are left as future research directions. Further, this ontology is developed before the acceptance of GDPR, and specific legislation requirements regarding data processing have not been considered.

Petova et al. (Petrova et al., 2017) propose and develop Financial Industry Business Ontology (FIBO) for contracts. It is a collection of eleven separate ontologies that define entities and processes in business and finance domains. FIBO does not focus on specific laws (e.g., GDPR). However, it provides a detailed semantic model of concepts such as contracts and agreements, which can be used as a foundation for any ontology focused on GDPR. Although FIBO does not focus explicitly on GDPR when modelling contracts, recent updates regarding its mapping to the legislation have been made. Furthermore, FIBO can be helpful for the formation of new ontologies that expect to depict business and monetary ideas and can be utilised in combination with the Data-Sharing Agreement Privacy Ontology (DSAP) (Li & Samavi, 2018) to assist information and interaction straightforwardness. We reused FIBO for many classes (e.g., *fibo-fnd-agr-ctr:MutualContractualAgreement* and *fibo-fnd-agr-ctr:Contract*) and properties (e.g., *fibo-fnd-agr-ctr:hasEffectiveDate* and *fibo-der-drc-ma:hasBeneficiary*) related to contracts.

3.2.3 Compliance Verification

Gangl (Matthias, 2019) analyses the impact of GDPR on third-party contracts. The author conducted a survey, which can be used for an in-depth analysis of contracting parties in the domain of cloud service providers. The survey's result is compared with the purpose of the GDPR to find out whether it supports the bilateral relationship in new and disruptive technologies. Further, they assess whether blockchain technology might be a valid alternative to achieve GDPR compliance. They argue that blockchain technology might be a valid alternative, but it has limitations.

Doe (Doe, 2018) describes guidelines for GDPR compliance verification from the perspective of the law firm sectors. The author provides a comprehensive introduction to the regulations and practicalities for law organisations in compliance with GDPR. The author makes a set of guidelines regarding the record of data processing, training needs, security, and contract documentation. There were only sets of guidelines for GDPR compliance verification, but there was no information about its implementation.

Ferrari (Ferrari, 2018) discusses data protection issues in blockchain technology. The author has examined different aspects of blockchain technology, which resonated or conflicted with the GDPR. For instance, GDPR is tailored to the model of centralised data storage. However, data stored on blockchains do not fall outside its application. The author emphasises GDPR requirements, which require more tension with the structure of blockchain technology (e.g., the right to the eraser, data minimisation, and conditions for transmission of data to third countries).

Starno et al. (Strano et al., 2009) present the implementation of a prototype for a contract compliance checker limited to B2B interactions. They describe the design and implementation of an independent third-party contract monitoring service (Contract Compliance Checker (CCC)), which provides the contract specification in force, and it is capable of observing and logging B2B interaction events while determining the business partner's consistency with contracts. They developed a contract specification language EROP (for Events, Rights, Obligations, and

Prohibitions) for the CCC. This model only deals with B2B and does not comply with GDPR.

Aziza et al. (Mamadolimova et al., 2011) present a contract compliance model for Islamic Finance Knowledge (IFK) using semantic web technology. Further, they describe contract compliance rule modelling to set Islamic Finance Contract (IFC) Heuristics that can be associated with a transaction model. Three comparative studies were conducted on the competing rules of formalism. This model does not focus on specific laws (e.g., GDPR).

Pantlin et al. (Pantlin et al., 2018) describe the attention on emerging market practices in supplier contracts in light of GDPR compliance. The authors discuss the complexity in the supply chain for businesses due to increased outsourcing to the cloud or the third-party external service providers. Further, challenges related to supplier contracts, such as rights audits, security measures, and sub-processors, are discussed as well. Therein, we do not have discussions and guidelines for GDPR compliance on B2B contracts.

Masoud and Omer (Barati & Rana, 2022) present a GDPR compliance tool supporting cloud providers in cloud-based service delivery. They introduce the encoding scheme for GDPR rules by creating legal questions, which is stored in the blockchain for auditing purposes. To investigate the execution cost of GDPR compliance checking, they deploy it on smart contracts in a blockchain test network. The presented GDPR compliance tool does not comply with B2B contracts and contracts without consent.

Maria et al. (Cambronero et al., 2017) presented an approach for the contract compliance evaluation regarding imperfect timing information to detect violation likelihood. They describe the importance of time constraints (e.g., a time window) for performing compliance on contracts. Based on these, they construct a time contract language. They only describe the model mathematically and do not provide any implementation details with any use case or tool.

To summarise, our work builds on the related work in the field and presents an exploration into (i) the construction and management of contracts; (ii) how KGs and ontologies can aid the building of common solutions and support knowledge discovery to ensure information reusability, reliability, and inference; and (iii) how the CCV with GDPR performs. The related work, overviewed in this section, is focused on exploring how organisations bind GDPR rights in contracts by providing guidelines (Doe, 2018), modelling the contracts (de Cesare & Geerts, 2012; Kabilan & Johannesson, 2003; Perrin & Godart, 2004; Petrova et al., 2017; Zou et al., 2010), developing contract compliance tools (Barati & Rana, 2022; Cambronero et al., 2017; Strano et al., 2009), discussing data protection issues in blockchain technology (Ferrari, 2018; Matthias, 2019), and describing contract management tools (Guo et al., 2021; Longo et al., 2015; Schmidt et al., 2019; Simić et al., 2021; Voronova, 2020). We followed the approach described in (Kabilan & Johannesson, 2003) to build our contract model by reusing FIBO (Petrova et al., 2017), which models standard contract-related classes and properties. For implementing the CCV, we followed the CCC approach in (Strano et al., 2009) and guidelines presented in (Doe, 2018).

3.3 Approach

This section details the approach of our study. However, before discussing our approach, we first provide an overview of the contract's lifecycle, as this provides the bigger picture of our work and its complexity. Following the overview of the contract lifecycle, in Section 3.3.1, we provide details on the semantic model that is used in contract KGs. In Section 3.3.2, we provide details about our approach to CCV, and finally, in Section 3.3.3, we provide example scenarios where CCV can be used. The CCV example scenarios are based on use cases UC1 and UC2 of the smashHit project, of which this work is part of.

Figure 3.1 presents the contract's lifecycle management, which describes the relationship between the CCV and contract management. The contract's lifecycle consists of six stages, as shown in Figure 3.1.

Contract request or offer is the initial stage of contract lifecycle management. In this stage, the initial contract draft is created in collaboration with different departments depending on the organisation. However, it is important to understand that most contracts are not agreed upon and signed as-is. There may be substantial changes that need to be made before all involved contractors can reach an agreement. In a B2C contract, a consumer must agree and sign the contract as-is to obtain benefits from online services, for example.

After the creation of the initial contract draft, the next stage is the negotiation stage. Here, the initial draft is available for all involved contractors to review. Often, this stage is the longest and most challenging stage in a contract's lifecycle management. Contractual parties' roles (e.g., the data subject, data controller, and data processor) are defined in this stage as well. A contract contains many contractual terms and clauses, which can be defined during the negotiation stage. Depending on the number of parties involved, it can take quite a bit of back-and-forth before a final agreement can be reached.

The next stage (i.e., signing) is responsible for the signing of the contracts. In this stage, contractors have to sign the contract once they agree on all contractual clauses defined in the previous stage. The contract management software often includes useful features that allow users to route the official version of the contract to contractors and allow individuals as needed during the contract signature process. Another important aspect of contract lifecycle management is the storage and execution of contracts. The execution of the contract begins once a contract is signed. It ensures that the contracts are properly filed, organised, and able to be found easily when needed. The complexity

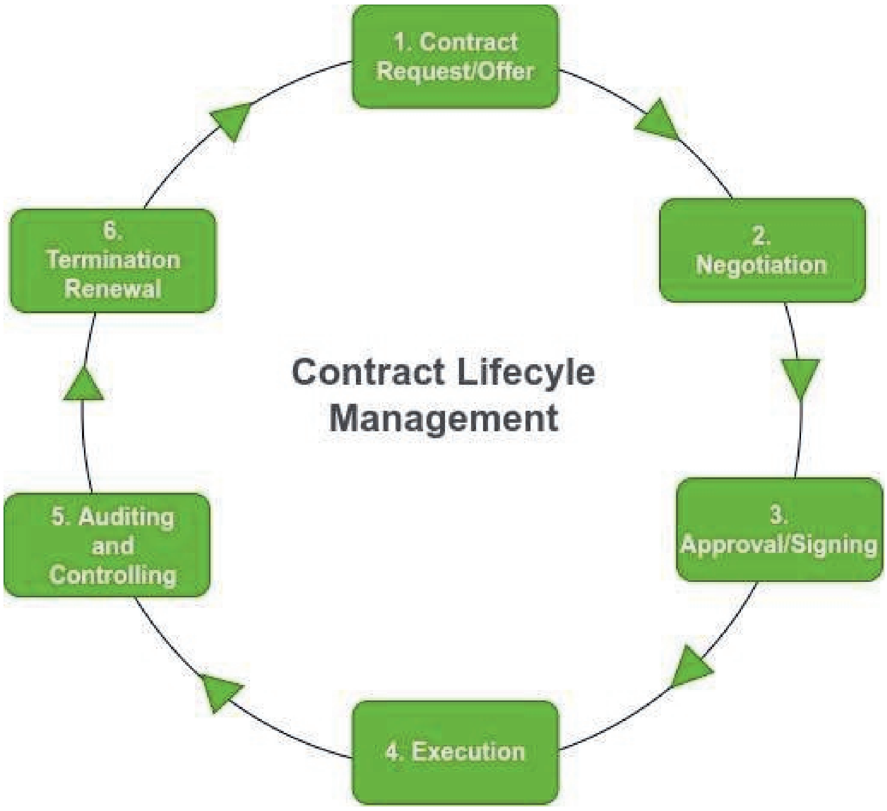

1. Contract Request/Offer
2. Negotiation
3. Approval/Signing
4. Execution
5. Auditing and Controlling
6. Termination Renewal
Contract Lifecycle Management

Figure 3.1: Contract lifecycle management.

of this stage increases while determining the contract's storage and contract execution. All contractual states (see details in Section 3.3.2) become valid at this stage.

After executing the contract, the next stage is auditing and controlling. It handles the contract audits and controls the execution of CCV checks and the contract's validity. Organisations and agencies need to focus on refining this stage and ensuring compliance so that nothing slips through the cracks. Furthermore, the termination or renewal stage handles the contract's status such that either the contract will be terminated or renewed. The CCV process is mainly focused on the auditing/controlling and termination/renewal stages of the contract's lifecycle management.

3.3.1 Semantic-Based Contract Model

In order to perform compliance verification checks, the CCV tool requires contractual information, such as contractual terms, contractors, and obligations. This information comes from data sources, such as KGs. This section presents how ontologies can be used as data models in KGs and how the semantic models of contracts are constructed. The smashHitCore ontology (Kurteva et al., 2023) is developed to perform GDPR compliance verification checks based on consent and contracts. In this paper, we only describe and present the semantic-based contract model, while the semantic representation of consent is presented in (Chhetri et al., 2022b).

The class *fibo-fnd-agr-ctr:Contract* (Figure 3.2) is used to model contracts and contractual obligations. In the context of use cases UC1 and UC2, a contract should present all of the necessary information for one to make an informed decision. However, the semantic model itself should also be generic enough so that it can be reused for various contracting scenarios. To cover both UC1 and UC2, we have reused *fibo-fnd-agr-ctr:MutualContractualAgreement* from FIBO (Petrova et al., 2017), which is generic enough to cover both use cases. A mutual contractual agreement involves an exchange of promises in which the promises made by each party represent considerations supporting the promises of the other party. Two categories of contracts have been modelled—*smashHitCore:BusinessToBusiness* and *smashHitCore:BusinessToConsumer*—according to UC1 and UC2.

A contract can be associated with a specific contractor via the object property *smashHitCore:hasContractors*. Specific terms and conditions can be related to a contract via the *smashHitCore:hasTerms* and *smashHitCore:hasObligations* object properties of class *fibo-fnd-agr-ctr:Contract*. Once a contract is signed (*smashHitCore:Signed*), the obligations associated with it become active (i.e., all contractors need to start adhering to them). If a contract has expired, then all obligations become invalid. To capture this information, we have modelled different obligation states with the class *smashHitCore:ObligationState*, namely *smashHitCore:Invalid*; *smashHitCore:Valid*; *smashHitCore:Pending*; and *smashHitCore:Fulfilled*. A contract has different object properties such as *fibo-fnd-agr-ctr:hasContractualElement* (e.g., terms and conditions) and *fibo-der-drc-ma:hasBeneficiary*. To differentiate between the date when a contract is created (i.e., all agents agree upon a set of terms and conditions and a policy) and the date when a contract becomes effective, we reused properties *smashHitCore:hasCreationDate* and *fibo-fnd-agr-ctr:hasEffectiveDate* accordingly. The property *smashHitCore:hasExpirationDate* refers to the date a contract expires, while *smashHitCore:hasEndDate* can be used in cases when a contract is terminated before its expiry date. To ensure the integrity of contracts, we defined the object property *smashHitCore:hasSignature*, which can be used to store the signatures of all contractors of a specific contract. Information about the used prefixes is available in Appendix 3.6. After presenting the semantic model of the contract, we now describe the CCV in Section 3.3.2.

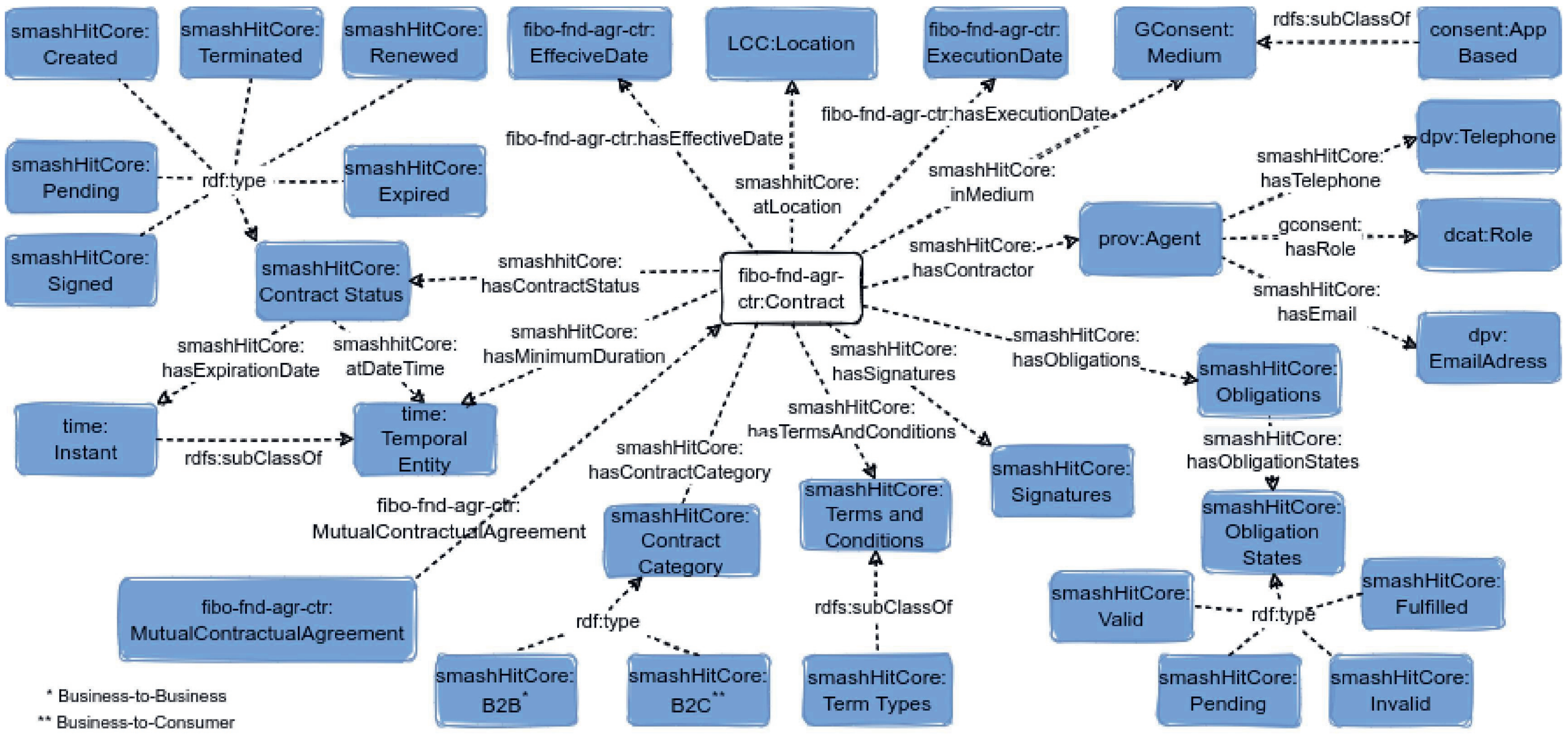

smashHitCore: Created
smashHitCore: Terminated
smashHitCore: Renewed
smashHitCore: Pending
smashHitCore: Expired
smashHitCore: Signed
rdf:type
smashHitCore: Contract Status
fibo-fnd-agr-ctr: EffeciveDate
fibo-fnd-agr-ctr:hasEffectiveDate
LCC:Location
smashhitCore: atLocation
fibo-fnd-agr-ctr: ExecutionDate
fibo-fnd-agr-ctr:hasExecutionDate
GConsent: Medium
smashHitCore: inMedium
rdfs:subClassOf
consent:App Based
dpv:Telephone
smashHitCore: hasTelephone
prov:Agent
smashHitCore: hasContractor
gconsent: hasRole
dcat:Role
smashHitCore: hasEmail
dpv: EmailAdress
fibo-fnd-agr-ctr:Contract
smashhitCore: hasContractStatus
smashHitCore: hasMinimumDuration
smashHitCore: hasExpirationDate
smashhitCore: atDateTime
time: Instant
rdfs:subClassOf
time: Temporal Entity
smashHitCore: hasObligations
smashHitCore: Obligations
smashHitCore: hasObligationStates
smashHitCore: Obligation States
smashHitCore: hasSignatures
smashHitCore: Signatures
smashHitCore: hasTermsAndConditions
smashHitCore: Terms and Conditions
rdfs:subClassOf
smashHitCore: Term Types
smashHitCore: hasContractCategory
smashHitCore: Contract Category
rdf:type
smashHitCore: B2B*
smashHitCore: B2C**
fibo-fnd-agr-ctr: MutualContractualAgreement
fibo-fnd-agr-ctr: MutualContractualAgreement
smashHitCore: Valid
rdf:type
smashHitCore: Fulfilled
smashHitCore: Pending
smashHitCore: Invalid
* Business-to-Business
** Business-to-Consumer

Figure 3.2: Semantic representation of contracts in smashHitCore.

3.3.2 CCV

Figure 4.1 shows a general overview of the CCV process, where a data source (e.g., KGs) is used as input. The first step in the CCV process is to extract contractual information from the data source. Data sources can vary and depend on the organisation. In the current scenario, we use KGs as a data source to store contractual information. We check the category of contracts in the second phase of the CCV process. The approach supports not only B2B and B2C contracts but also consent-based contracts. In the third phase, consent-based validation performs on each contract. In CCV, a validation check is performed to validate the contractual clause in the fourth phase to obtain violation or expiration results. These results are presented in the fifth phase of the CCV process. The contract status and clause state are updated in the KGs with violation or expiration results. In the last phase of the CCV process, contract violation or expiration notifications are sent to contractual parties.

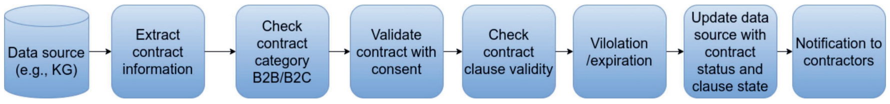

Data source (e.g., KG)
Extract contract information
Check contract category B2B/B2C
Validate contract with consent
Check contract clause validity
Vilolation /expiration
Update data source with contract status and clause state
Notification to contractors

Figure 3.3: A general overview of CCV process.

After discussing the general overview of the CCV, we now describe the relationship of contract breaches with GDPR based on UC1 and UC2. To illustrate, let us assume we have two organisations: LexisNexis and Infotripla Oy. The first acts as a data controller, whereas the second acts as a data processor according to GDPR. In data processing, where a contract is required, it must satisfy the requirements defined by GDPR (e.g., Art. 28, 32). For instance, the data processor must notify the data controller if there is a breach of a contract. The insurers can view the information about the data storage and its usage. Personal information needs to be anonymised by the tool. The tool must also satisfy the TOMs defined by the GDPR (see detail in Section 3.4.2.1). Complying with GDPR compliance, our tool fulfils all these requirements. It performs compliance verification checks, ensuring a contract breach, control over all running contracts, updating the contractual parties about the contract statuses, and GDPR compliance verification.

The contract dates (i.e., start date, effective, and end date), status, and clause states are key factors in performing contract validity checks. These contract dates comprise creation, effective or execution, and end date. The value of contract status depends on changes in contract dates and clause states values. These clause states are associated with contractual clauses. In the negotiating process, these states become active once a contract is signed. We explore contractual clauses and the clause states to illustrate contract breaches. As an example, we can write a contractual clause as a tuple, as shown as follows:

$$clause(s, a, o, [ts, te])$$

where s = subject; a = action; o = object; ts = start time; and te = end time.

The contractual clause states comprise *Invalid*, *Fulfilled*, *Pending*, and *Violated* (Irwin et al., 2006). A contractual clause becomes invalid if the end time is already passed when it is assigned. The contractual clause is said to be fulfilled if it has been assigned and its action has been carried out its activities during the time window [ts, te]. If a contractual clause has been assigned, has not been fulfilled, and is not invalid but has an end time that is passed, then it is violated. If a contractual clause is not invalid and has not yet become fulfilled or violated, then it is pending. We explore such a clause with the following example. Suppose we have the following.

$$cl_1 = obl(Bob, submitreview(Bob, p1), [16/02/22, 25/02/22])$$
$$cl_2 = obl(Bob, submitreview(Bob, p2), [16/02/22, 25/02/22])$$

There are two contractual clauses in tuples cl_1 and cl_2, where Bob has to submit two reviews for the papers p1 and p2 within a specified time framework. If Bob submits his review for p1 on 25 February, cl_1 becomes fulfilled. If the tuple cl_2 has started, its status becomes pending until 25 February 2022. Tuple cl_2's status becomes violated if Bob does not submit a review for p2 on 25 February 2022. A contract has many contractual terms that define contractual clauses. Contract status changes due to these contractual clause states. A contractual clause with violations also changes the contract's (associated with that clause) status to violate. The tool sends a notification about these violations to the contractual parties. We formalise the logic we discussed here in order to make the presentation of rules more clear and more concise. For the sake of simplicity, we omitted existential quantifiers for unbound variables. Hence, we have obtained the following rules: (i) The first rule states that a clause will have a pending state if it does not have any of the other clause states, such as fulfilled, invalid, or violated; (ii) the second rule states that if a clause is pending and has an associated obligation that is set in the past, then the clause is automatically set to invalid; (iii) the third rule states that if a clause with a pending state that has an associated obligation which has not been submitted until the end date, then it is violated; (iv) the fourth rule states the opposite compared to the previous rule, in the sense that a pending state becomes fulfilled if all associated obligations are submitted within the required period; finally, (v) the last rule states that if a clause is violated, then the corresponding contract is also violated.

$$
\begin{aligned}
&\forall X \forall Y\ Contract(X) \ \wedge\ hasClause(X,Y) \ \wedge\ Clause(Y) \\
&\wedge\ \neg hasState(Y, fulfilled) \ \wedge\ \neg hasState(Y, invalid) \ \wedge\ \neg hasState(Y, violated) \\
&\rightarrow hasState(Y, pending) \\
&\forall X \forall Y\ Contract(X) \ \wedge\ hasClause(X,Y) \ \wedge\ Clause(Y) \ \wedge\ hasState(Y, pending) \\
&\wedge\ hasObligation(Y, S, A, O, time_s, time_e) \ \wedge\ time_{assign} > time_e \rightarrow hasState(Y, invalid) \\
&\forall X \forall Y\ Contract(X) \ \wedge\ hasClause(X,Y) \ \wedge\ Clause(Y) \ \wedge\ hasState(Y, pending) \\
&\wedge\ hasObligation(Y, S, A, O, time_s, time_e) \ \wedge\ time_{curr} > time_e \rightarrow hasState(Y, violated) \\
&\forall X \forall Y \forall S \forall A \forall O \forall time_s \forall time_e\ Contract(X) \ \wedge\ hasClause(X,Y) \ \wedge\ Clause(Y) \\
&\wedge\ hasState(Y, pending) \wedge\ hasObligation(Y, S, A, O, time_s, time_e) \ \wedge\ time_{curr} <= time_e \\
&\rightarrow hasState(Y, fulfilled) \\
&\forall X\ Contract(X) \ \wedge\ hasClause(X,Y) \ \wedge\ Clause(Y) \ \wedge\ hasState(Y, violated) \\
&\rightarrow hasState(X, violated).
\end{aligned}
$$

Let us consider a running contract c_1 with the associated clause cl_1. After the application of the first rule, we obtain the following.

$$Contract(c_1) \ \wedge\ hasClause(c_1, cl_1) \ \wedge\ Clause(cl_1) \rightarrow hasState(cl_1, pending)$$

Given $time_{curr} = 27/2/2022$, and after the application of the third rule, we obtain the following.

$$
\begin{aligned}
&Contract(c_1) \ \wedge\ hasClause(c_1, cl_1) \ \wedge\ Clause(cl_1) \ \wedge\ hasState(cl_1, pending) \\
&\wedge\ hasObligation(cl_1, bob, submitreview(bob, p_1), p_1, 16/2/22, 25/2/22) \\
&\wedge\ 27/2/22 > 25/2/22 \rightarrow hasState(cl_1, violated)
\end{aligned}
$$

Finally, given the last rule from above, for the contract, we also deduce the following.

$$
\begin{aligned}
&Contract(c_1) \ \wedge\ hasClause(c_1, cl_1) \ \wedge\ Clause(cl_1) \ \wedge\ hasState(cl_1, violated) \\
&\rightarrow hasState(c_1, violated) \quad \square
\end{aligned}
$$

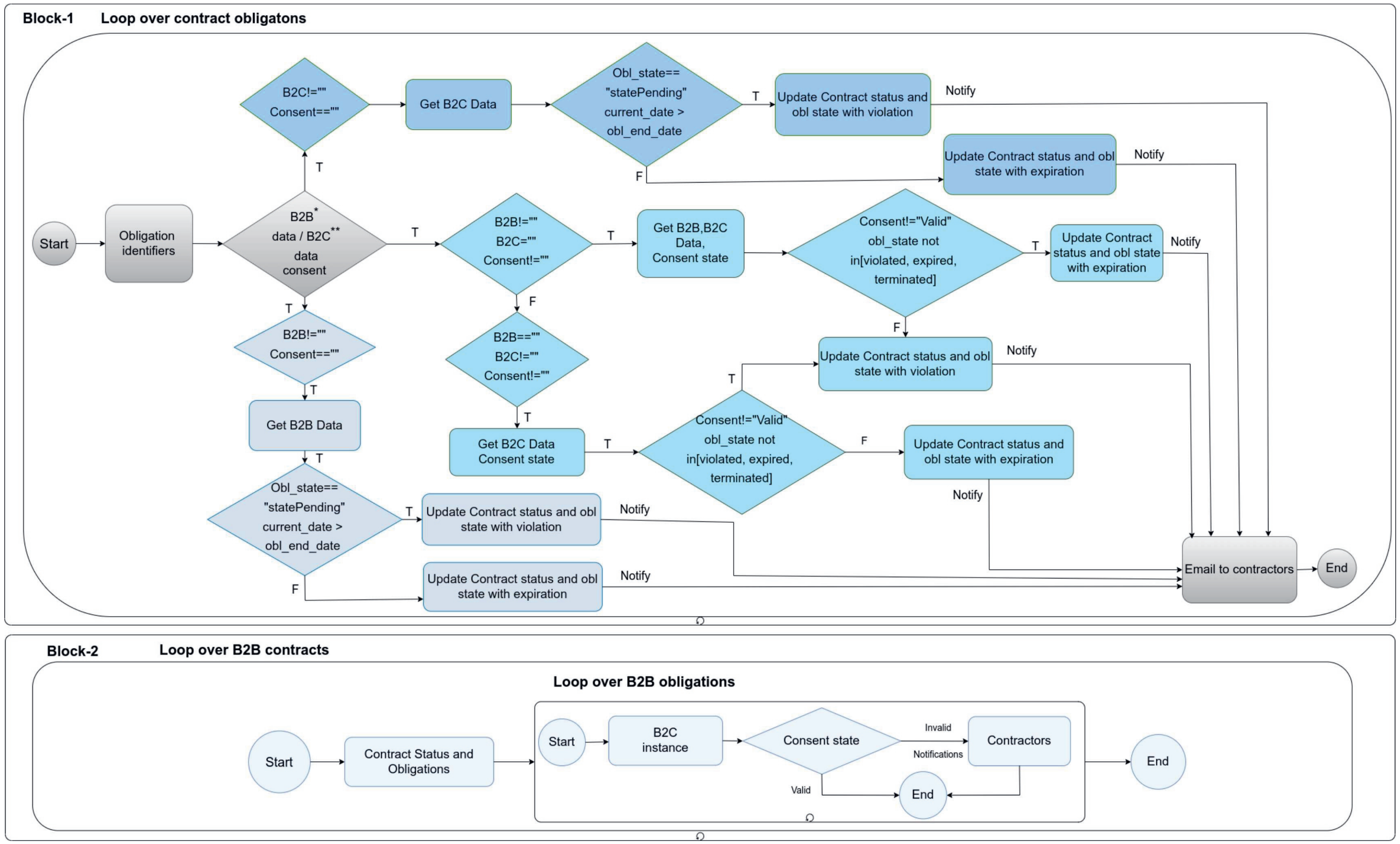
After an illustration of a contract breach with examples and rules, we describe CCV process scenarios in Section 3.3.3.

3.3.3 CCV Scenarios

This section presents the CCV overview with four scenarios based on UC1, UC2, and industrial requirements discussed in Section 4.1. Before discussing the scenarios, we present the overview of the CCV process (see Figure 3.4), which shows a distinction between each scenario with a specific colour. Second, we describe each scenario in more detail in Sections 3.3.3.1–3.3.3.3.

Figure 3.4 presents the overall CCV process comprising scenarios, such as B2C or B2B contracts, consent-based B2C contracts, and consent-based compliance verification on B2B contracts. It presents the extraction of all contractual clauses from the KGs via the SPARQL (Simple Protocol and RDF (Resource Description Framework) Query Language) (SPARQL Working Group, 2013) endpoints. A contract repository may have multiple contractual clauses that need multiple iterations for validation. Each contractual clause has a contractual party, *contractID*, *stateID*, *termID*, and a time window (including start and end time). The contractual clause state and contract status require validating each contractual clause. In CCV, only contracts having created or signed statuses and contractual clauses with a pending state are involved. The information about a contract's status and contractual states is extracted from the KGs. With this information and the current date, we can validate contractual clauses.

Figure 3.4 is divided into two blocks, namely block-1 and block-2. Block-1 shows the following three scenarios used to perform CCV: (i) B2C contract; (ii) B2B contract; (iii) Consent-based B2C and B2B contract. While in block-2, we show the CCV with the fourth scenario (i.e., automatic detection of contract breaches based on informed consent, where the consent has expired and the contracts—based on that consent—are still running). In smashHit, a B2C contract is created between an insurer (acts as a data subject) and LexisNexis (acts as a data controller). While a B2B contract is made between LexisNexis (acts as a data controller) and an organisation (as a data processor). To make a clear distinction among all four scenarios, we assigned different colours to scenarios. In block-1, the B2C contract scenario is presented in baby blue colour, the B2B contract scenario is in iceberg colour, and the informed-consent base contract scenario is in fresh air colour. Alice blue colour is assigned to present the fourth scenario of the CCV in block-2. In the following subsections, we present each of them in more detail.



Block-1
Loop over contract obligatons
Start
Obligation identifiers
B2B* data / B2C** data consent
T
B2C!="" Consent==""
Get B2C Data
Obl_state== "statePending" current_date > obl_end_date
T
Update Contract status and obl state with violation
Notify
F
Update Contract status and obl state with expiration
Notify
B2B!="" B2C="" Consent!=""
T
Get B2B,B2C Data, Consent state
Consent!="Valid" obl_state not in[violated, expired, terminated]
T
Update Contract status and obl state with expiration
Notify
F
Update Contract status and obl state with violation
Notify
B2B=="" B2C!="" Consent!=""
Get B2C Data Consent state
Consent!="Valid" obl_state not in[violated, expired, terminated]
Update Contract status and obl state with expiration
Notify
B2B!="" Consent==""
Get B2B Data
Obl_state== "statePending" current_date > obl_end_date
Update Contract status and obl state with violation
Notify
Update Contract status and obl state with expiration
Notify
Email to contractors
End
Block-2
Loop over B2B contracts
Loop over B2B obligations
Start
Contract Status and Obligations
Start
B2C instance
Consent state
Invalid
Notifications
Contractors
Valid
End
End

Figure 3.4: CCV process overview.

3.3.3.1 B2C or B2B Contracts

The first two scenarios are very similar, with the difference being in the contract's category. Both must satisfy the following conditions for compliance verification: (i) a B2B or B2C contract and (ii) the consent information must be empty. If the condition result is evaluated as *true*, contractual information of the B2B or B2C is extracted from the KGs via the SPARQL endpoint. Based on this information, another conditional check is performed on the contractual clause with its states and end date. If this is the case, the condition result is true, and the contract status and its clause states are updated in the KGs with violation information. Otherwise, both contract status and clause states update with expiration information. In both cases, the tool sends notifications automatically to contractual parties with a violation or expiration information.

3.3.3.2 Consent-Based Compliance Verification on B2C Contracts

The third scenario is based on consent, which has two parts: B2C contract and B2B contract. In the first part, the conditions (*B2B*=="" and *B2C!*="" and *Consent!*="") must be true to perform compliance verification checks on B2C contracts. Once the condition is *true*, the tool extracts B2C contract information from the KGs. Based on this information, the tool validates it with the consent state and the state of the contractual clause. If the condition result is *true*, the tool updates the contract's status and the contractual clause states with violation information. In a case where the condition's result is *false*, the contract status and contractual clause states will update with expiration information. The tool notifies the contractual parties automatically based on this violation or expiration information. This process repeats for the second part, which is B2B contract compliance verification. In a case where the consent state is invalid, it must also expire all contracts associated with that consent. This process repeats and executes until there is no clause left for validation.

3.3.3.3 Consent-Based Compliance Verification on B2B Contracts

The tool also supports detecting contract violations based on consent automatically. To illustrate it, we consider a data subject (e.g., a person and software) possessing a B2C contract with LexisNexis (i.e., data controller) and consenting to share data for five months. Based on this contract, LexisNexis makes a B2B contract with an organisation ABC (i.e., data processor) to obtain benefits for selling the data. For illustration purposes, let us assume that the data subject revoked consent after two months. The contracts associated with that consent must be terminated or expire. Since consent has been revoked, the B2C contract must also be terminated or must expire. The B2B contract, which is created based on a B2C contract, must also be terminated or expire. Let us assume for any reason that the B2B contract is still running. In that case, how will the data subject know about this contract breach? Our tool supports detecting this type of breach automatically and updates the data subjects about each contract associated with them. We present the process of this scenario in block-2 of Figure 3.4.

In the process of Block-2, we extract all B2B contracts with their contractual clause information. To create a consent-based B2B contract, we have a B2B contract clause possessing a B2C contract reference. With the help of this reference, we can extract the consent state from a B2C contract. Furthermore, we perform a compliance check with the consent state and contract status. If the consent state is invalid and the contract status has not ended or expired, then the tool sends notifications to the data subjects about this violation.

3.4 Architectural Design and Implementation

This section details the architectural design of the CCV, which is presented in Section 3.4.1 and its implementation details are presented in Section 3.4.2.

3.4.1 CCV Architectural Design

Figure 3.5 presents the CCV architectural design. It follows a micro-services architecture pattern. A micro-services architecture pattern is one in which all modules are cohesive, independent processes that interact through messages (Dragoni et al., 2017). The Service Layer is a key component, which comprises the Core, the Resources, the Application Programming Interface (API) Layer, and the contract compliance scheduler. We describe each of them in the following subsections.

3.4.1.1 Core

The Core module is divided into two sub-modules: Data Processing and Shared Services. The first one is responsible for data management to support required operations, such as contract creation, auditing or controlling, and compliance verification checking. The query processor and storage are two sub-components for supporting data processing operations. The query processor component contains the SPARQL queries required to deal with contracts (e.g., contract data in KGs), while the storage module handles the query processor's execution. Shared services include modules that assist other modules in their operations, such as contract and compliance verification checks.

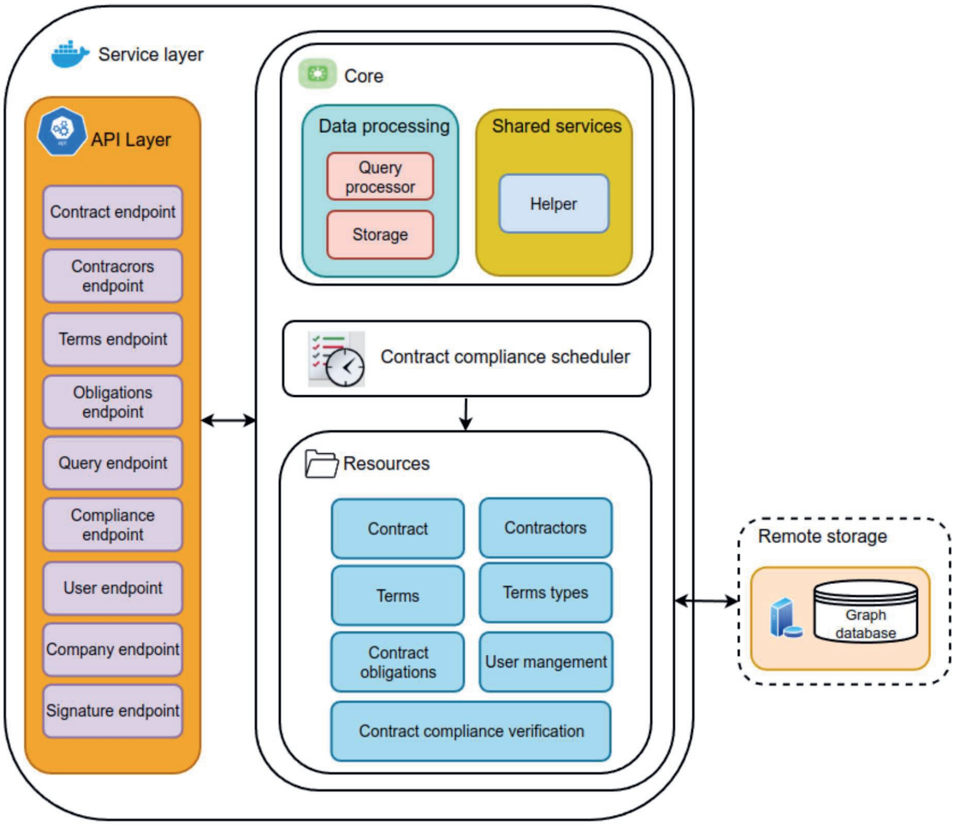


Figure 3.5: Architectural design of the automated GDPR compliance verification tool.

3.4.1.2 API Layer

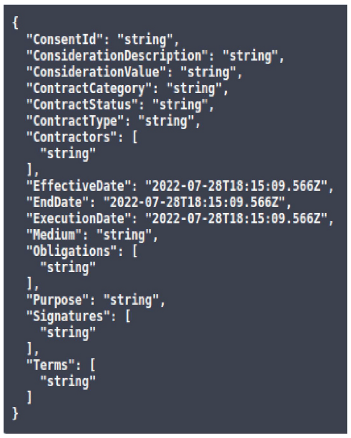
The API layer is used to interact with the CCV tool. It provides access to the compliance verification tool's functionalities via REST (REpresentational State Transfer) endpoints. Contract search, contractual parties management, contract audit, and contract compliance are the core features of the API layer.

3.4.1.3 Resources

The Resources component is a part of the Service Layer, which contains classes, such as Contract, Contractual Parties, Contractual Terms, Contractual Clause Types, Contractual Clause, and Contract Compliance. These are required for the management and compliance verification of contracts. Each class has Create, Read, Update, and

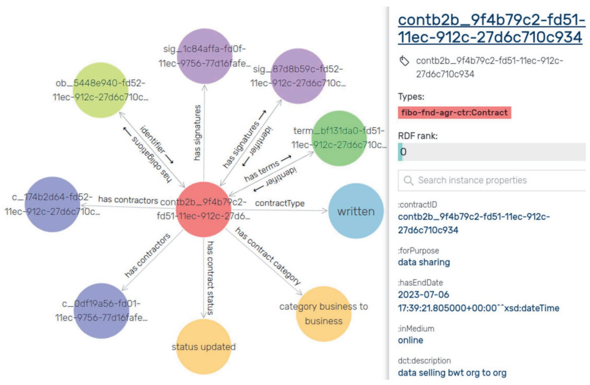
Delete (CRUD) operations. Search by ID (e.g., contract id, contractual party's id, and clause id) and searching the bulk of records are two common types of search. The contract compliance component is responsible for performing automated compliance verification checks. Figure 3.6 presents the JSON (JavaScript Object Notation) schema and the semantic representation of a B2B contract.

Figure 3.6 represents an overview of a contract instance from our knowledge graph and all information related to it. The centre node (in red colour) represents an instance of the class *fibo-fnd-agr-ctr:Contract*. This instance's label has been encrypted for security and privacy reasons. All other nodes represent entities related to that specific contract instance. For example, a contract can have several contractors associated with it (see the nodes connected to the contract instance via the "has contractors" relationship).



```
{
  "ConsentId": "string",
  "ConsiderationDescription": "string",
  "ConsiderationValue": "string",
  "ContractCategory": "string",
  "ContractStatus": "string",
  "ContractType": "string",
  "Contractors": [
    "string"
  ],
  "EffectiveDate": "2022-07-28T18:15:09.566Z",
  "EndDate": "2022-07-28T18:15:09.566Z",
  "ExecutionDate": "2022-07-28T18:15:09.566Z",
  "Medium": "string",
  "Obligations": [
    "string"
  ],
  "Purpose": "string",
  "Signatures": [
    "string"
  ],
  "Terms": [
    "string"
  ]
}
```

(a)



sig_1c84affa-fd0f-11ec-9756-77d16fafe...
sig_87d8b59c-fd52-11ec-912c-27d6c710c...
ob_5448e940-fd52-11ec-912c-27d6c710c...
term_bf131da0-fd51-11ec-912c-27d6c710c...
identifier
has obligations
has signatures
has signatures
identifier
has terms
identifier
c_174b2d64-fd52-11ec-912c-27d6c710c...
has contractors
contb2b_9f4b79c2-fd51-11ec-912c-27d6...
contractType
written
has contractors
has contract status
has contract category
c_0df19a56-fd01-11ec-9756-77d16fafe...
category business to business
status updated

contb2b_9f4b79c2-fd51-11ec-912c-27d6c710c934

contb2b_9f4b79c2-fd51-11ec-912c-27d6c710c934

Types:
fibo-fnd-agr-ctr:Contract

RDF rank:
0

Search instance properties

:contractID
contb2b_9f4b79c2-fd51-11ec-912c-27d6c710c934

:forPurpose
data sharing

:hasEndDate
2023-07-06 17:39:21.805000+00:00^^xsd:dateTime

:inMedium
online

dct:description
data selling bwt org to org

(b)

Figure 3.6: A snapshot of the JSON schema used for contract and the contract module's creation (or representation) of contract in the legal KG. (**a**) Contract JSON schema. (**b**) KG representation of a contract in GraphDB.

3.4.1.4 *Remote Storage*

For the construction and management of contracts, Ontotext GraphDB (Sirma Group, 2019) is used, which supports the RDF and SPARQL. The SPARQL endpoint is used to perform CRUD operations on contracts, while RDF is used in the construction of semantic models of contracts.

3.4.1.5 Contract Compliance Scheduler

The Flask APScheduler ("Flask-apscheduler", 2022) is used to handle time-based job scheduling tasks for automated detection of contract breaches. For instance, we set up a scheduling task for contract breach detection, which executes every day at 01:00 AM. The data controller makes compliance verification checks directly by calling the contract compliance endpoint through the contract's REST APIs endpoints. The source code of the entire process can be found on GitHub (Tauqeer, 2021).

3.4.1.6 Contract REST API

For interactions with the tool, the API Layer implements REST endpoints. For an ideal representation of the API documentation, swagger (SmartBear, 2018) is used. The swagger REST APIs endpoints for contracts can be found on GitHub (Tauqeer, 2021), which requires performing CCV checks and managing contracts. We divide the contract REST APIs endpoints into seven parts: contracts; contractual parties; clause types; contractual terms; contract contractors (contractual parties, which are associated with a particular contract); contractual clauses; contract signatures; and contract compliance. Each part in terms of functionality is able to perform CRUD operations. For binding requests from swagger API for KG, custom contract schema are used as shown in Figure 3.6. This contract schema comprises basic information (e.g., contract category, contract types, and purpose) and collections of contractual parties, contractual terms, contractor signatures, and contractual clauses.

3.4.2 Implementation

This section details the tool's implementation based on the use cases and industrial requirements discussed in Section 4.1. The implementation of TOMs is presented in Section 3.4.2.1. We describe the libraries used for this implementation in Section 3.4.2.2, while in Section 3.4.2.3, we present the implementation details for each component of the tool.

3.4.2.1 The Implementation of TOMs

Performing the CCV with GDPR, our solution follows the "data protection by design and by default" principle. For this principle, implementing TOMs is a key requirement according to GDPR (Art. 25 (1)). The adoption of internal policies (Rec. 78) states that it is the responsibility of the data controller (or data processor) to implement TOMs, ensuring that processing is performed under GDPR (Art. 4 (7), Art. 24). Our tool implements the following TOMs.

Data Encryption

The first TOM relates to confidentiality (Art. 32 (1) (a)) to encrypt the processing data. For encryption, we use the deterministic searchable encryption technique. Two algorithms the Rivest–Shamir–Adleman (RSA) (Ízdemir & Ídemiş Ízger, 2021) with Public Key Cryptography (PKCS) Standards and asymmetric Advanced Encryption Standard (AES) (Selent, 2010) are used for this purpose. Further, by implementing authentication procedures and identity management, only registered components have access to endpoints.

Protection Against External Influences on Systems and Services

This TOM relates to Art. 32 (1) (b), which is defined as "the ability to ensure the ongoing confidentiality, integrity, availability, and resilience of processing systems and services" to ensure that the systems and services are planned correctly and according to the intended purpose. The tool implements security measures (e.g., authentication procedures see detail in Section 3.4.2.3) and user-based access to prevent unauthorised data access.

Documentation of Data Syntax

The third TOM relates to the documentation of data, its availability and resilience (Art. 32 (1) (b)). The entire code follows the Python Enhancement Proposals (PEP)-8 (Van Rossum et al., 2001) coding convention and is commented for better understandability. For an ideal representation of the API documentation, Swagger (SmartBear, 2018) is used.

Reduction in Non-Required Attributes of Data Subjects

Our fourth TOM is used to enable data minimisation according to GDPR (Art. 32 (1) (d), 25 (1)). Our tool implements the data minimisation requirements according to GDPR by establishing retention periods (e.g., dates) for personal data processing to ensure GDPR contract compliance verification. For example, our contract's REST APIs endpoint creation defines a minimal set of variables, such as purpose and dates (execution date and end date for retaining the data only for as long as it is necessary to fulfil the purpose of processing).

Role Concepts with Graduated Access Rights Based on Identity Management and a Secure Authentication Process

This TOM relates to the purpose of limitation according to (Art. 32 (1) (d), 25 (1))), which is about testing, assessing and evaluating the effectiveness of TOM to ensure the security of data processing. It takes the purpose of the limitation into account and defines permissible purpose changes. Our tool uses the JavaScript Object Notation (JSON) and Web Tokens (JWT) (Jones et al., 2010) based access control. Furthermore, user-based access on endpoints is implemented.

Translating legal requirements into technical implementations is not easy. The Standard Data Protection Mode (SDM) ("SDM", 2020) provides appropriate measures, transforming the GDPR legal bases to qualify for TOMs. A summary of GDPR requirements mapped with their data protection goals is described in ("SDM", 2020) (Table in Section C2, p. 28). The SDM is as follows:

1. Systematises data protection requirements in the form of protection goals;
2. Systematically derives generic measures from protection goals, supplemented by a catalogue of reference measures;
3. Systematises the identification of risks in order to determine protection requirements of the data subjects resulting from the processing; and
4. Offers a procedure model for modelling, implementation, and continuous control and testing of processing activities.

3.4.2.2 System Setup for Evaluation

We summarise the libraries and software that were used in this implementation in Table 5.1. We selected these libraries because of our tool's requirements. For instance, GraphDB was selected due to having capabilities, such as more intuitive data visualisation, storage, and management. The free edition of GraphDB is not sufficient for simultaneous queries because it does not support concurrency or parallelism of more than two queries. In order to alleviate this issue, the Enterprise Edition (EE) of GraphDB can be deployed instead. A Docker container in a system with 32 GB (gigabyte) random-access memory (RAM), a 1.7 gigahertz (GHz) AMD Ryzen 7 PRO 4750U processor, and 1 terabyte (TB) storage is used for deploying the service layer. Linux with variant distributions, such as Ubuntu and Debian, is used for all deployment setups.

3.4.2.3 Automated GDPR CCV Tool Implementation

This section provides the implementation details of each component of the CCV tool, such as the API layer and CCV layers.

API Layer

The main functionality of the API layer is to implement the REST endpoints for contracts. It enables user-based access as demanded by GDPR's integrity and confidentiality principle (Art. 5 (1) (f)) by custom JWT implementation. All the contract's REST API endpoints are only accessed through a valid JWT token, which is created upon

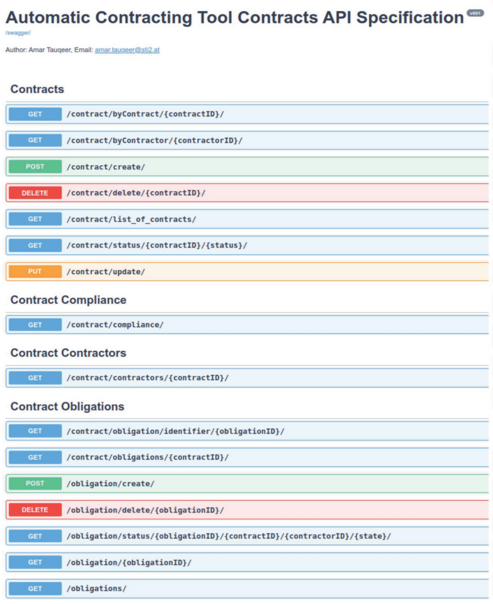
Table 3.1: List of software (or libraries) that were used in the implementation.

Software/Libraries	**Version**
Python (“Python”, 2021)	3.8
Flask (“Flask”, 2021)	1.1.2
Flask-RESTful (“Flask-RESTful”, 2021)	0.3.8
Flask-SQLAlchemy (“Flask-SQLAlchemy”, 2022)	2.5.1
Python Requests	2.25.1
Flask Apispec (“Flask-Apispec”, 2018)	0.11.0
Pycryptodome (“PyCryptodome”, 2014)	3.10.1
SPARQLWrapper (“SPARQLWrapper”, 2008)	1.8.5
Docker ((“Docker”, 2013) Community Edition)	20.X
SQLite	2.6
GraphDB free edition (Sirma Group, 2019)	9.4.1
Protégé	5.5.0
Pyjwt (“PyJWT”, 2015)	1.7.1

successful login. Furthermore, the standard REST practices, such as OpenAPI Specification (OAS) version 2.0 and swagger, are used in implementing the API layer for describing the contract’s REST endpoints (see Figure 3.7).

Data Processing

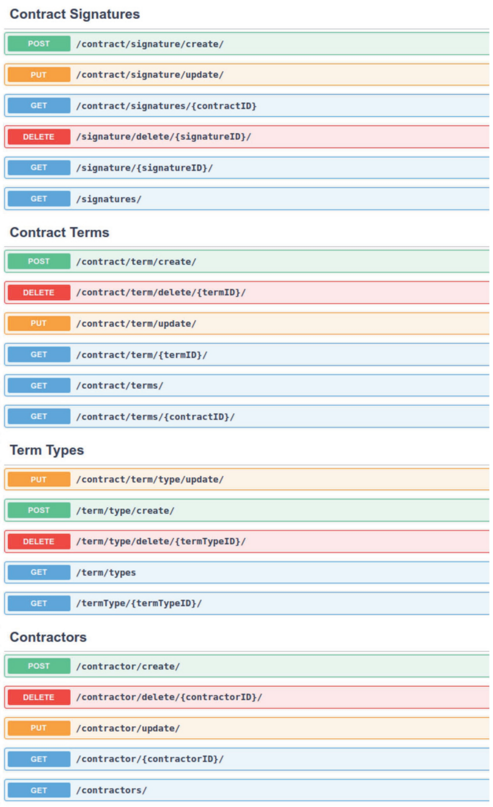
The data processing module comprises predefined SPARQL queries with the contractual information to be filled in during the run-time. These queries are organised based on the Resource (see detail in Section 3.4.1.3) component of the CCV tool. In Figure 3.8, we present a snapshot of the SPARQL query, which is used to extract all information with respect to a contract such as contract contractors, terms, obligations, and contract category. It contains two functions *prefix* and *get_all_contractors*. All namespaces are stored in the first one required for the execution of the SPARQL query, while the second one stores the SPARQL query.



Automatic Contracting Tool Contracts API Specification v001
/swagger/
Author: Amar Tauqeer, Email: amar.tauqeer@sti2.at

Contracts
```
GET     /contract/byContract/{contractID}/
GET     /contract/byContractor/{contractorID}/
POST    /contract/create/
DELETE  /contract/delete/{contractID}/
GET     /contract/list_of_contracts/
GET     /contract/status/{contractID}/{status}/
PUT     /contract/update/
```
Contract Compliance
```
GET     /contract/compliance/
```
Contract Contractors
```
GET     /contract/contractors/{contractID}/
```
Contract Obligations
```
GET     /contract/obligation/identifier/{obligationID}/
GET     /contract/obligations/{contractID}/
POST    /obligation/create/
DELETE  /obligation/delete/{obligationID}/
GET     /obligation/status/{obligationID}/{contractID}/{contractorID}/{state}/
GET     /obligation/{obligationID}/
GET     /obligations/
```

(a)



Contract Signatures
```
POST    /contract/signature/create/
PUT     /contract/signature/update/
GET     /contract/signatures/{contractID}
DELETE  /signature/delete/{signatureID}/
GET     /signature/{signatureID}/
GET     /signatures/
```
Contract Terms
```
POST    /contract/term/create/
DELETE  /contract/term/delete/{termID}/
PUT     /contract/term/update/
GET     /contract/term/{termID}/
GET     /contract/terms/
GET     /contract/terms/{contractID}/
```
Term Types
```
PUT     /contract/term/type/update/
POST    /term/type/create/
DELETE  /term/type/delete/{termTypeID}/
GET     /term/types
GET     /termType/{termTypeID}/
```
Contractors
```
POST    /contractor/create/
DELETE  /contractor/delete/{contractorID}/
PUT     /contractor/update/
GET     /contractor/{contractorID}/
GET     /contractors/
```

(b)

Figure 3.7: Contract REST API's endpoints in Swagger. (**a**) Part 1. (**b**) Part 2.

Shared Service

Two functions *function_map* and *list_to_query* are implemented by the shared services module as shown in Figure 3.9. The other modules, such as data processing and compliance verification can use these shared services. The *list_to_query* function is used to convert the array of JSON inputs into the SPARQL query format for supporting contract creation activities by the contract module, while the *function_map* is used to perform the mapping to the actual function.

Contract Compliance Scheduler

The Flask APScheduler ("Flask-Apispec", 2018) is used to handle time-based job scheduling tasks for automated detection of contract breaches. For example, the tool sets up a scheduling task for contract breaches detection, which executes every day at 01:00 AM. Furthermore, the data controller makes a compliance verification check

```python
def prefix(self):
    prefix = textwrap.dedent("""PREFIX : <http://ontologies.atb-bremen.de/smashHitCore#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX dc: <http://purl.org/dc/elements/1.1/>
        PREFIX dpv: <http://www.w3.org/ns/dpv#>
        PREFIX prov: <http://www.w3.org/ns/prov#>
        PREFIX dcat: <http://www.w3.org/ns/dcat#>
        PREFIX fibo-fnd-agr-ctr: <https://spec.edmcouncil.org/fibo/ontology/FND/Agreements/Contracts/>
        PREFIX dct: <http://purl.org/dc/terms/>
    """)
    return prefix

def get_all_contracts(self):
    query = textwrap.dedent("""{0}
        select *
        where{{
        ?Contract rdf:type fibo-fnd-agr-ctr:Contract;
            :contractID ?contractId;
            :hasContractStatus ?contractStatus;
            :hasContractCategory ?contractCategory;
            dct:identifier ?consentId;
            :forPurpose ?purpose;
            :contractType ?contractType;
            fibo-fnd-agr-ctr:hasEffectiveDate ?effectiveDate;
            fibo-fnd-agr-ctr:hasExecutionDate ?executionDate;
            :hasEndDate ?endDate;
            :inMedium ?medium;
            dct:description ?consideration;
            rdf:value ?value .
    }}
    """).format(self.prefix())
    return query
```

Figure 3.8: A snippet of code from the query processor module.

directly by calling the contract compliance endpoint through the contract's REST API endpoint. Figure 3.10 presents a compliance verification scheduling task based on the current date.

Resources

The Resource component of CCV implements sub-modules, such as Contract, Contractual Parties, Contractual Terms, Contractual Clause Types, Contractual Clause, and Contract Compliance, which are used for the management and to perform compliance verification checks on contracts. Each class has its procedures for performing CRUD operations. For instance, creating a new contract requires contract data in JSON format following JSON schema, as shown in Figure 3.7. This scheme is used to transform the contract data into KG and is validated with marshmallow ("Marshmallow", 2013-2023). Marshmallow is a framework-agnostic library for converting complex data types, such as objects, to and from native Python data types.

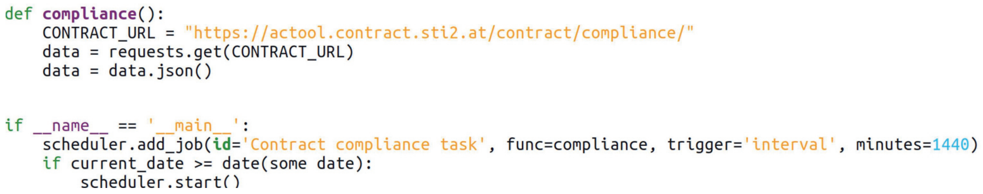
Each component performs the following similar functionalities: (i) extracting all details (records) of the component; (ii) extracting component-specific details (based on IDs e.g., contractID, contractorID); (iii) component creation; (iv) updating a particular component; and (v) deleting a specific component. All components of the Resource module perform partial and full auditing. Figure 3.11 presents a snapshot of a contract's partial and full audit in the JSON Schema. The basic information of the contract, such as

```
def function_map(self, name):
    """ Map to actual function
    :param name: name which function to map
    :return: function name
    """
    mapfunc = {
        "get_all_contracts": self.get_all_contracts,
        "get_contract_by_contractor": self.get_contract_by_contractor,
        "get_contract_by_provider": self.get_contract_by_provider,
        "get_contract_by_id": self.get_contract_by_id,
        "get_signature_by_id": self.get_signature_by_id,
        "get_contractor_by_id": self.get_contractor_by_id,
        "get_company_by_id": self.get_company_by_id,
        "get_all_contractors": self.get_all_contractors,
        "get_all_companies": self.get_all_companies,
        "get_all_terms": self.get_all_terms,
        "get_all_signatures": self.get_all_signatures,
        "get_contract_signatures": self.get_contract_signatures,
        "get_term_type_by_id": self.get_term_type_by_id,
        "get_term_by_id": self.get_term_by_id,
        "get_obligation_by_id": self.get_obligation_by_id,
        "get_all_obligations": self.get_all_obligations,
        "get_contract_obligations": self.get_contract_obligations,
        "get_all_term_types": self.get_all_term_types,
        "get_contract_terms": self.get_contract_terms,
        "get_contract_contractors": self.get_contract_contractors,
        "get_contract_compliance": self.get_contract_compliance,
        "contract_update_status": self.contract_update_status,
        "get_obligation_identifier_by_id": self.get_obligation_identifier_by_id,
        "get_signature_identifier_by_id": self.get_signature_identifier_by_id,

    }
    return mapfunc[name]

def list_to_query(self, data, whatfor):
    """ Convert list to query
    :input: list
    :returns: SPARQL query string
    """
    querydata = ""
    for vlaue in data:
        strs = ":" + whatfor + " :" + vlaue + ";\n"
        querydata = strs + querydata
    return querydata
```

Figure 3.9: A snippet of code from the helper module.

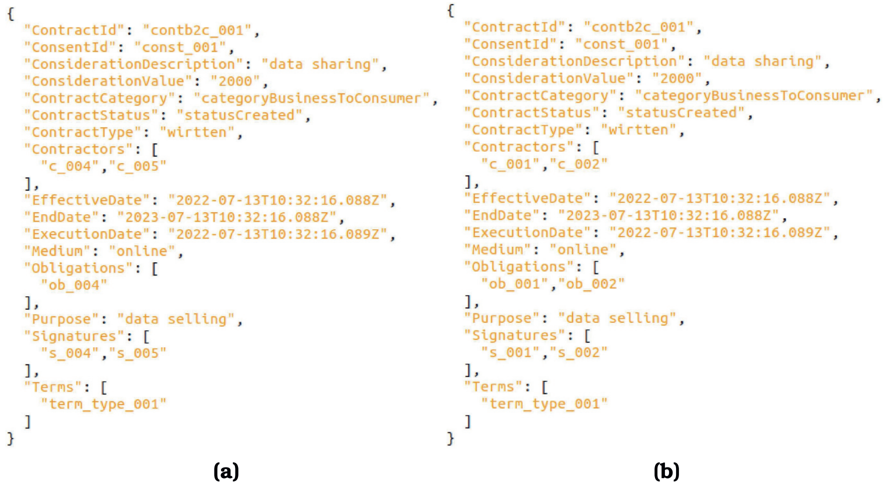
contract category and contract dates (e.g., start, effective, and execution), is provided for the partial contract audit, as shown in Figure 3.11. While Figure 3.11 presents a full contract audit information in JSON Schema. It contains not only the basic information of the contract but also other contractual information, such as a collection of contractual parties, a collection of contractual terms, and a collection of contractual clauses.

The CCV (see detail in Section 3.3.2) implements an automated compliance verification



```
def compliance():
    CONTRACT_URL = "https://actool.contract.sti2.at/contract/compliance/"
    data = requests.get(CONTRACT_URL)
    data = data.json()


if __name__ == '__main__':
    scheduler.add_job(id='Contract compliance task', func=compliance, trigger='interval', minutes=1440)
    if current_date >= date(some date):
        scheduler.start()
```

Figure 3.10: A snippet of code for scheduling the compliance verification check.



```
{
  "ContractId": "contb2c_001",
  "ConsentId": "const_001",
  "ConsiderationDescription": "data sharing",
  "ConsiderationValue": "2000",
  "ContractCategory": "categoryBusinessToConsumer",
  "ContractStatus": "statusCreated",
  "ContractType": "wirtten",
  "Contractors": [
    "c_004","c_005"
  ],
  "EffectiveDate": "2022-07-13T10:32:16.088Z",
  "EndDate": "2023-07-13T10:32:16.088Z",
  "ExecutionDate": "2022-07-13T10:32:16.089Z",
  "Medium": "online",
  "Obligations": [
    "ob_004"
  ],
  "Purpose": "data selling",
  "Signatures": [
    "s_004","s_005"
  ],
  "Terms": [
    "term_type_001"
  ]
}
```

(a)

```
{
  "ContractId": "contb2c_001",
  "ConsentId": "const_001",
  "ConsiderationDescription": "data sharing",
  "ConsiderationValue": "2000",
  "ContractCategory": "categoryBusinessToConsumer",
  "ContractStatus": "statusCreated",
  "ContractType": "wirtten",
  "Contractors": [
    "c_001","c_002"
  ],
  "EffectiveDate": "2022-07-13T10:32:16.088Z",
  "EndDate": "2023-07-13T10:32:16.088Z",
  "ExecutionDate": "2022-07-13T10:32:16.089Z",
  "Medium": "online",
  "Obligations": [
    "ob_001","ob_002"
  ],
  "Purpose": "data selling",
  "Signatures": [
    "s_001","s_002"
  ],
  "Terms": [
    "term_type_001"
  ]
}
```

(b)

Figure 3.11: A snapshot of contract partial and full audit JSON Schema. (**a**) Partial audit. (**b**) Full audit.

check to perform on contracts. This compliance check performs only on active contracts (i.e., a contract having status, such as created, pending, and updated). The CCV implementation is based on four scenarios discussed in Section 3.3.3. In the first two scenarios, the implementation is based on B2C and B2B contracts. The third scenario is based on consent-based contracts. The fourth scenario performs the compliance checks on B2B contracts, where the consent has expired but the contracts (associated with that consent) are still running. Each component's implementation details with respect to the Resource module can be found on GitHub (Tauqeer, 2021).

Security

Two algorithms RSA (Ízdemir & Ídemiş Ízger, 2021) and AES (Selent, 2010) are used to ensure secure data processing in the CCV tool. The RSA algorithm's proven capability and security robustness over the last 30 years is a valid reason for its selection. While considering the de facto standard for symmetric encryption and standardised by the National Institute of Standards and Technology (NIST) as an encryption technique, the AES algorithm is selected, which is fast and secure (Federal Information Processing Standards Publication (FIPS), PUB, 2022). The function *key_generate* is used to create and export the public and private keys using RSA. For data encryption and decryption, the security module implements two functions *rsa_aes_encrypt* and *rsa_aes_decrypt*. The Public-Key Cryptography Standards (PKCS) # 1 OPTIMAL ASYMMETRIC ENCRYPTION PADDING (OAEP) (Garg & Yadav, 2014) padding scheme is used by the RSA's implementation, which is defined by RFC 8017. To encrypt and decrypt the keys for symmetric encryption algorithms, RSA is used. The complete implementation details can be found on GitHub (Tauqeer, 2021).

3.5 Evaluation

This section presents the evaluation of our tool with a focus on performing CCV compliance functionalities. It is based on tools' key functionalities, such as contract creation, contract audit, and CCV checks. Both use cases (UC1 and UC2) require scalable solutions to handle end users. In Section 3.5.1, we present the CCV performance evaluation, while we show the TOMs evaluation in Section 3.5.2. Furthermore, for CCV implementation and performance evaluation, we use the system's setup, as described in Section 3.4.2.2.

3.5.1 CCV Performance Evaluation

To evaluate the CCV performance, we created ten different contract instances based on UC1 and UC2 and measured their execution times. The process repeats to create instances of contract terms and contractual clauses. The contract creation process is divided into five parts: (i) the contract's basic information, (ii) contractors, (iii) contract terms, (iv) contractor signatures, and (v) contractual clauses. The execution time of a contract creation is based on the total execution time of all the above five parts. For this performance evaluation, the contract's information is provided manually. For this performance evaluation, the information about the instances of contract, contractual terms, contractual clauses, and terms types can be found on GitHub (Tauqeer, 2021) (see contract-creation.ods file in the evaluation folder).

Before discussing evaluation results, we introduced terms, such as *contract create an instance (CTI)*, *contract audit (CTA)*, *contract terms create (CTT)*, *contract terms audit (CTTA)*, *contract obligation create (CTO)*, *contract obligation audit (CTOA)*, and *COMP* for the instances of contract creation, contract audit, contract terms, contract obligations, and contract compliance. Figure 3.12 represents the contract creation (including five parts) instances (*CTI1*, *CTI2*, ... *CTI10*) on the x-axis, while the execution time (in minutes) of each contract creation instance shows on the y-axis. On the right side of Figure 3.12, we have contract audit instances (*CTA1*, *CTA2*, ... *CTA10*) on the x-axis and execution time (in minutes) on the y-axis in the Figure 3.12. Similarly, in Figure 3.13, we can see the evaluation results of contract terms create and audit with their instances. In addition, Figure 3.14 represents the evaluation results of contractual clauses creation and audit with their instances, where instances are shown on the x-axis and execution time on the y-axis. Finally, we present the compliance verification results in terms of execution time in Figure 3.15, where the instances (*COMP1*, *COMP2*, ... *COMP10*) are presented on the x-axis and the execution time (in seconds) on the y-axis.

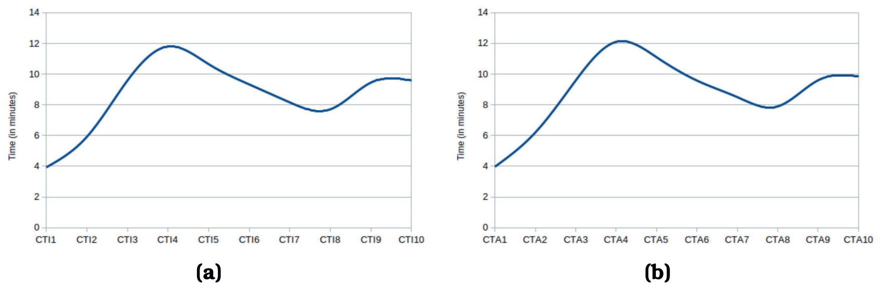


Figure 3.12: Performance evaluation on contract creation and audit. (**a**) Time spent on contract creation. (**b**) Time spent on contract audit.

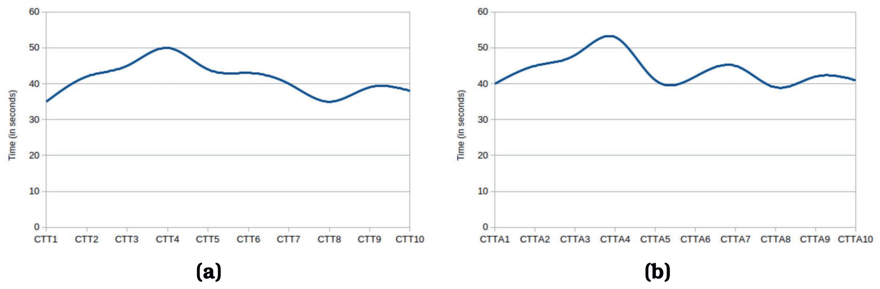


Figure 3.13: Performance evaluation on contract term creation and audit. (**a**) Time spent on contract term creation. (**b**) Time spent on contract term audit.

The minimum time spent on a contract creation process is 3.55 min, while 11.48 is the maximum time spent on a contract as shown in Figure 3.12. The *CTI4* took 11.48 min because it has two contract terms and four contractual clauses, whereas *CTI1* only has a contract term and a contractual clause. Similarly, the maximum time of 4.50 min was spent on contractual clauses (four clauses), 2.50 min on contract (two terms), 0.42 s on contractor signatures (three signatures), 2.00 min on contractual parties (two contractors), and 1.40 min on the contract's basic information. Contract audit and creation processes have taken almost the same execution time because both have

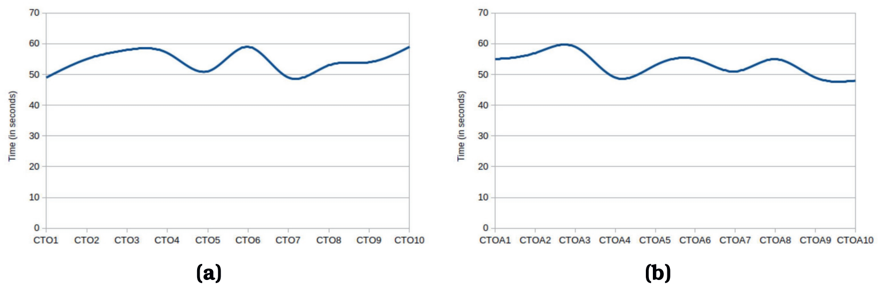


Figure 3.14: Performance evaluation on contract clause creation and audit. (**a**) Time spent on contractual clause create. (**b**) Time spent on contractual clause audit.

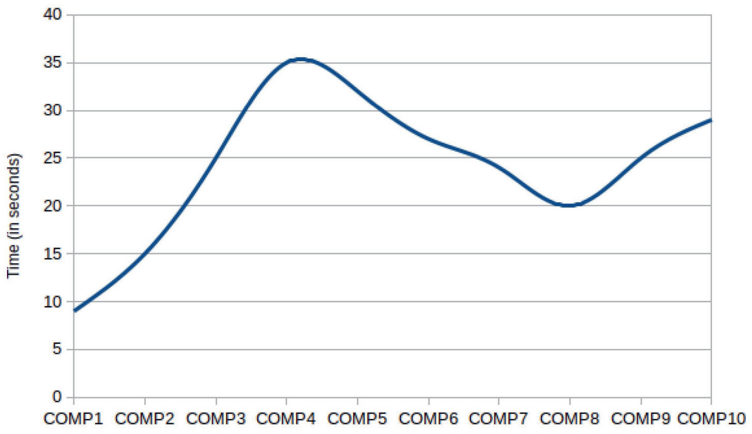


Figure 3.15: Time spent on CCV.

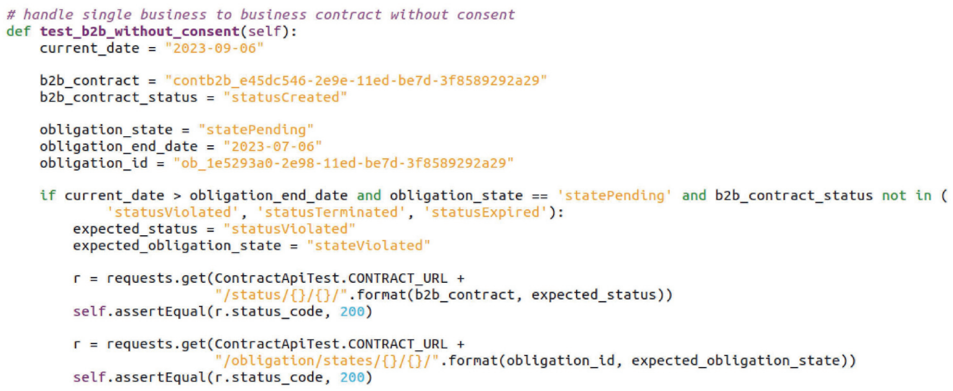
the same contents as shown in Figure 3.12a,b. We did not consider partial creation and audits here.

We also measured the performance evaluation on the contract's term and contractual clause, which are shown in Figures 3.13a,b and 3.14a,b. The average time spent on the contract term creation or audit is 0.40 s, as shown in Figure 3.13a,b, while the creation or audit of the contractual clause took an average of 0.55 s, as shown in Figure 3.14a,b. Based on contract creation instances depicted in Figure 3.12, we measured the time spent on compliance verification in Figure 3.15. It shows a strong relationship with contract instances, and if the instances take more time, the compliance verification will also take more time. For example, the *COMP4* (compliance instance related to *CTI4*) took 36 s to complete, whereas *COMP8* (related to *CTI8*) took only 20 s. More information about the contract evaluation performance is shown in Table 4.3 (the fastest and slowest measurements in terms of time are highlighted in bold). The contract contents (e.g., terms, and obligations) can also affect the execution time of the contract creation, audit, and compliance. The encryption and decryption of the information can result in higher time in performance evaluation. However, these are also required to increase security measures. These extra time-consuming activities are associated with compliance verification and causes the CCV tool to slow down.

To evaluate the correctness of the CCV tool, we performed unit tests with 28 different test case scenarios. The evaluated test cases include the CRUD operations relating to contracts, such as contract terms and contractual obligations. Moreover, the test cases also include the CCV tool's compliance verification operations. The CCV compliance verification unit test cases includes 5 different test scenarios described in Section 3.3.3. Figure 3.16 shows a code snippet of the unit test for the B2B contract scenario without consent. As shown in Figure 3.16, the test case takes the contract's ID, status, current date, obligation state, obligation end date, and obligation ID for compliance verification. Further, a condition (i.e., *current date > obligation end date and obligation state = 'Pending' and b2b contract status not in ('Violated', 'Terminated', 'Expired')*) is checked. The contract status and obligation state must be updated by the tool if the condition result yields true.

Table 3.2: Contract performance evaluation.

ID	Contract Basic Information (Time in Minutes)	Contractual Parties (Time in Minutes)	Contractual Term (Time in Minutes)	Contractual Clauses (Time in Minutes)	Contractor Signatures (Time in Minutes)	Total (Time in Minutes)
1	**1.00**	**0:50**	**0.16**	**0.44**	**0.16**	**3.55**
2	1.05	1:55	0.32	1.58	0.30	5.54
3	1.20	**2:00**	1.50	3.50	0.35	9.34
4	1.30	1:58	**2.50**	**4.50**	0.40	**11.48**
5	**1.40**	**2:00**	1.50	3.52	0.37	10.39
6	1.20	1:57	1.48	3.40	0.35	9.33
7	1.30	**2:00**	1.20	2.40	0.40	8.16
8	1.10	1:58	1.30	2.30	0.35	7.42
9	1.25	**2:00**	1.40	3.45	0.37	9.45
10	1.35	1:55	1.58	3.25	**0.42**	9.34



```python
# handle single business to business contract without consent
def test_b2b_without_consent(self):
    current_date = "2023-09-06"

    b2b_contract = "contb2b_e45dc546-2e9e-11ed-be7d-3f8589292a29"
    b2b_contract_status = "statusCreated"

    obligation_state = "statePending"
    obligation_end_date = "2023-07-06"
    obligation_id = "ob_1e5293a0-2e98-11ed-be7d-3f8589292a29"

    if current_date > obligation_end_date and obligation_state == 'statePending' and b2b_contract_status not in (
            'statusViolated', 'statusTerminated', 'statusExpired'):
        expected_status = "statusViolated"
        expected_obligation_state = "stateViolated"

        r = requests.get(ContractApiTest.CONTRACT_URL +
                         "/status/{}/{}/".format(b2b_contract, expected_status))
        self.assertEqual(r.status_code, 200)

        r = requests.get(ContractApiTest.CONTRACT_URL +
                         "/obligation/states/{}/{}/".format(obligation_id, expected_obligation_state))
        self.assertEqual(r.status_code, 200)
```

Figure 3.16: A unit test code snippet for B2B test scenario to check the CCV correctness.

[20]The unit test cases with their results logs, and the source code are available on GitHub (Tauqeer, 2021). Our evaluation of unit test cases demonstrates that the CCV tool performs the intended tasks, such as compliance verification and contract creation correctly.

3.5.2 TOMs Evaluation

In Section 3.4.2.1, we summarised and evaluated five data protection goals associated with TOM regarding GDPR (data processing) for contracts. We evaluated the contracts module manually. We discussed the manual evaluation in Section 3.5.1. For instance, the contractual parties' personal data are encrypted first and then stored in the KG, which is related to GDPR (Art. 32 (1) (a)) confidentiality (see detail in Section 3.4.2.1). Similarly, the PEP-8 coding convention is used for the documentation of data syntax. For the automated procedure, we wrote test use cases and executed them using Python's unit test framework (Python Software Foundation, 2021). We have different conditions to make these tests. For example, in testing the endpoint *GetContractContractor*, we provide the contract number. In the case of providing the contract number, the test case returns a success response. The test case returns a failed response in case of missing the contract number. More information about each test case can be found on GitHub (Tauqeer, 2021).

3.6 Conclusions and Future Work

By building on our previous work in (Chhetri et al., 2022b), in this paper, we presented our CCV (Contract Compliance Verification) tool for digital contract management based on knowledge graphs. To be specific, we presented an approach for automated CCV checks over contracts. Further, we discussed factors that must consider the technical requirements to satisfy industry requirements as discussed in Section 4.1. The CCV tool has a micro-services architecture and utilises an ontology and a knowledge graph, which support the interoperability of data.

The CCV tool supports contract management, based on consent, with B2C (Business-to-Consumer) and B2B (Business-to-Business) contracts that can be generalised to other domains. Our tool supports not only consent-based contract generation but also considers scenarios in which consent is not required (see Section 3.4). We evaluated the CCV tool with the performance and scalability regarding contracts. We also evaluated the CCV methods' correctness by performing unit test cases. This research is conducted in collaboration with legal experts and industrial partners. It can help SMEs (small and medium enterprises) in binding GDPR (General Data Protection Regulation) legal bases with data sharing contracts. The CCV tool improves both the compliance verification process and the contract lifecycle. Future studies include (i) improving the signing process with digital signatures; (ii) implementing digital licensing on contracts using DALICC (Pellegrini et al., 2018)/Licence Clearance Tool (LCT) ("License Clearance Tool", 2021); (iii) improving the negotiation process where the data subject will have more options to collaborate on making contract clauses; (iv) performing validation to graph-based data using Shapes And Constraints Language (SHACL); (v) extracting the existing contracts (e.g., paper contracts and unstructured contracts) from external data sources to translate into RDF (Resource Description Framework); and (vi) optimising the performance of the tool.

Acknowledgements

This research is supported by the smashHit EU project funded under Horizon 2020 (grant number 871477). We express our gratitude to Friederike Knoke and Samuel Iheanyi Nwankwo from Leibniz Universität Hannover (LUH) Institut für Rechtsinformatik (IRI) for the legal analysis of the contracts for our use cases. Furthermore, we thank our industry collaborators LexisNexis Risk Solutions, Volkswagen AG, Infotripla and Forum Virium Helsinki for supporting the use cases for our work. We also thank our colleagues Rainer Hilscher and Antonio J. Roa-Valverde from the University of Innsbruck (UIBK) for participating in the discussions and providing appropriate feedback on our work.

Author Contributions: Conceptualisation, A.T. and A.K.; methodology, A.T.; software, A.T. and T.R.C.; validation, A.T., T.R.C. and A.A.; formal analysis, A.T. and A.K.; investigation, A.T., A.A. and T.R.C.; resources, A.T.; data curation, A.T.; writing—original draft preparation, A.T.; writing—review and editing, A.T., A.K., A.A., T.R.C. and A.F.; visualisation, A.T.; supervision, A.F.; project administration, A.F.; funding acquisition, A.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Horizon 2020 project smashHit (grant number 871477).

Data Availability Statement: Not applicable

Conflict of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

Semantic models prefix

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix gconsent: <https://w3id.org/GConsent#> .
@prefix dpv: <http://www.w3.org/ns/dpv#> .
@prefix fibo-fnd-agr-ctr:
  <https://spec.edmcouncil.org/fibo/ontology/FND/Agreements/Contracts/> .
@prefix smashHitCore: <http://ontologies.atb-bremen.de/smashHitCore#> .

@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix LC: <https://www.omg.org/spec/LCC/Countries/CountryRepresentation/> .
```

CHAPTER 4

An Integrated Approach to GDPR-compliant Data Sharing Employing Consent, Contracts, and Licenses

This chapter is based on:

Tauqeer A., Chhetri T.R., Fensel A., David R., and Ahmeti A. An Integrated Approach to GDPR-compliant Data Sharing Employing Consent, Contracts, and Licenses (Submitted to Data & Knowledge Engineering journal, under review.)

Abstract

The GDPR outlines six legal bases necessary to lawfully process personally identifiable data. Presently, research predominantly focuses on consent and contracts, limiting options for data sharing, particularly when multiple legal bases are needed. For instance, while one may consent to data sharing, one might wish to impose usage restrictions, necessitating a license. Combining these bases is complex due to the need for a comprehensive understanding of each and the challenge of designing a compliant and functional system. In response, our paper introduces a semantic-based approach and tool enabling GDPR-compliant data sharing across multiple legal bases: consent, contracts, and licenses. Expanding on our prior GDPR Contract Compliance Verification (CCV) tool, which facilitated consent and contract-based sharing, we now incorporate licenses. This extension involves utilizing SHACL validations for compliance checks, securing contract signings with digital signatures, implementing SHACL repairs for automatic data inconsistency fixes, and performance evaluations. The paper showcases the effectiveness of SHACL and how it enhances the tool for GDPR-compliant data sharing across multiple legal bases, as demonstrated through performance testing.

Keywords— Data sharing, Contracts, GDPR compliance, Ontology, Knowledge graphs, SHACL, License, Digital signatures.

4.1 Introduction

In today's digitized world, data sharing has a significant impact on almost every type of industry. In particular, data sharing has enabled numerous opportunities in industries such as healthcare, smart cities, insurance, and autonomous vehicles (Cai et al., 2023; Cao et al., 2015). One of further examples where data sharing played a vital role is the case of the COVID-19 contract tracing application, which helped in containing the spread of the virus and thus saving lives (Ahmed et al., 2020). Another example is in smart cities, where data sharing is necessary to enable smart city components such as smart mobility and a smart environment (Savithramma et al., 2022). However, as important as data sharing is in today's digital world, there has been a major concern about privacy and misuse of the data.

This increase in privacy concerns has led to the adoption of different privacy measures, particularly, in legislation. One of the notable examples of privacy legislation is the European Union General Data Protection Regulation (GDPR) which was adopted on May 25, 2018. GDPR defines the six legal bases under which processing of personally identifiable data can be done and applies to everyone processing the data of an EU citizen. One needs to obtain "consent" or "the right to erasure" from the data subject (DS) beforehand to be able to process any personally identifiable data. In further cases, such as the selling of data between companies, a contract "a written or spoken or online agreement between contractors such as DS, data controller (DC), and data processor (DP)" is also required. Moreover, when one wants to legally secure third-party digital assets such as datasets, software, or contents, a license "is an official permission or permit to do, use, or own something (as well as the document of that permission or permit)[1]" is required. Licenses are not part of GDPR but have legal bases, included in the EU law[2]. Licenses are extremely important because they enable businesses or individuals to retain control of their work or intellectual property while still benefiting from it. For example, an individual or small business that owns intellectual property can profit by licensing it to other individuals or businesses. Therefore, to enable seamless data sharing and secure the digital, one needs to deal with multiple (different) legal bases.

A number of studies have been conducted on contracts, consent, and licenses (Chhetri et al., 2022b; Havur et al., 2018; Semantha et al., 2023; Tauqeer et al., 2022). However, challenges remain due to the siloed manner in which these works are conducted, thereby limiting data sharing options and potential benefits. In particular, two major challenges exist: (i) understanding the legal basis and implementing it systematically in a manner that is legally compliant; (ii) combining the isolated works to build an integrated system (or tool) that is practically applicable and performant. Moreover, a similar challenge has also been highlighted by Zafar et al. (Zafar et al., 2018), according to whom 50% of projects face data integration challenges. Additionally, an

[1]https://en.wikipedia.org/wiki/License
[2]https://ec.europa.eu/isa2/solutions/european-union-public-licence-eupl_en/

integrated system provides a comprehensive platform that can be utilized for multiple data sharing options as needed, thereby eliminating the need to utilize multiple platforms and thus offering users convenience, especially in terms of usability and time savings. For example, consent is insufficient for online services and necessitates contracts (Tauqeer et al., 2022). If there is no integrated system (or tool) that offers options for both consent and contract, then one needs to use different platforms separately. As a result, users (e.g., industry) are burdened by the need to keep track of multiple tools and integrate and coordinate them—overcoming this motivates our work.

Many use cases in the smart city and insurance domains face the same data sharing difficulties and GDPR legal basis requirements as discussed above. *Smart City Services* (UC1[3]) and *Insurance Services* (UC2[4]) are two particular use cases in the smashHit[5] project that support this study in building GDPR legal bases requirements for data sharing in smart city and insurance domains. The smashHit project's aim was to create a scalable, trustworthy, and secure system for GDPR-compliant data sharing and management of consent and contracts in the connected automobile and smart city domains. We derive key requirements from both use cases for enabling GDPR-compliant data sharing including consent and contracts in smart cities and insurance domains. In particular, UC1 focuses on data sharing in the insurance domain where data sharing is a key to informed decisions. However, informed consent is the primary legal basis for data sharing between the DS and DP (e.g., insurance company), when data are sold to third parties, additional terms and conditions must be presented and agreed upon. These serve as the foundation for a contract, which serves as the primary legal basis. In contrast, UC2 depicts a data sharing scenario in the smart city sector in which an unprecedented amount of data (contractual information) is emitted, shared, and analyzed by different agents (i.e., software, humans, and organizations). A contract is also required in circumstances such as Business-to-Business (B2B) data sharing in UC2 since it specifies each contract party's obligations as well as the particular terms and conditions that must be followed. Similarly, besides contracts and consent, the creation of a license based contract "contains not only a standard license but also provides freedom to contractors to attach additional data sharing requirements in the form of contractual clauses to have (more) control over data" for data sharing in the smashHit project can lead to another potential benefit in addition to GDPR-complaint legal bases.

Another fascinating scenario is an application "City Feedback System" in the UC1 of the smashHit project, where the objectives of the application are: (1) collecting traffic and environmental feedback from end users in a city environment, (2) collecting usable data from other available sources (e.g., Volkswagen[6] (VW) vehicle sensor data) to

[3]https://smashhit.eu/d6-5-demonstrator-of-services-using-integrated-cpp-and-insurance-data/

[4]https://smashhit.eu/d7-5-demonstrator-of-services-using-integrated-traffic-smart-city-and-cpp-data/

[5]https://cordis.europa.eu/project/id/871477

[6]https://www.vw.com/en.html

be utilized in the services including Feedback application, (3) study and contribute to an external platform to manage consents between Infotripla[7] and data owners, (4) to show the feedback and other collected data on a publicly available application, and (5) forwarding applicable feedback to e.g., city maintenance department for corrective actions. The collected vehicle sensor data then can be shared with other smashHit parties/companies (e.g., Forum Virium Helsinki[8]) using the data traceability component under a data processing/sharing contract (i.e., a GDPR-complaint legal basis and a complex process) between smashHit and VW. One possible solution to avoid this complex process is to associate standard licenses (e.g., through the DALICC (Pellegrini et al., 2019) framework) with data sources under the GDPR-complaint data-sharing contract. This reduces the complexity of the process, cost, and effort of not writing contractual clauses with complete descriptions (as these are described in the DALLIC framework already). Instead, one can use the license IDs (with integration of the DALICC REST APIs endpoints) directly from the DALICC framework using REST APIs–also a motive for this research. The process of association of licenses with digital assets is described in Section 4.5.2.

Another typical scenario is academic data sharing, where data sharing advantages for researchers can be corpulent as they are associated with high risks while sharing personally identifiable data or sensitive data such as name, email, phone, address (Fecher et al., 2015; Urovi et al., 2022). Due to the nature of data sharing and its challenges, the landscape of rights and licensing resources has become complicated. Sam Grabus and Jane Greenberg (Urovi et al., 2022) addressed the following questions in academic data sharing: where is the best place for a researcher to learn about facilitating the complex process of rights management? and which standardized licenses would be most appropriate for sharing a particular type of data, and which metadata standards and ontologies can help address these needs?

Last but not least another interesting scenario is UC2 in the smashHit project, where a car's GPS data (latitude, longitude, fuel, mileage, etc.) can be used with a standard license for data sharing before consenting or entering into a data sharing contract. Without using licenses, car drivers have to give information with full descriptions whenever they grant consent or create a data sharing contract. It is a time and effort-consuming process. On the other hand, the use of standardized licenses (e.g., using the DALICC framework) helps car drivers save time and effort by not putting information with full descriptions when granting consent or creating data sharing contracts while driving cars. Instead, they can use standard licenses with GDPR-complaint contracts for data sharing.

Therefore, in an effort to address the aforementioned challenges and enhance our previous work, automated contract compliance verification (CCV) (Tauqeer et al., 2022), we present our integrated approach. In particular, we make the following improvements to our previous work: (i) making the contract management and compliance

[7]https://futuremobilityfinland.fi/member/infotripla/
[8]https://forumvirium.fi/en/

verification tasks more semantically compliant using SHACL (SHACL Working Group, 2017) (Shapes Constraint Language), a W3C (World Wide Web Consortium) recommendation for validating knowledge graphs (KGs) (Rabbani et al., 2022), as opposed to the previous Pythonic[9] way of performing validation, (ii) in addition to GDPR legal bases such as contracts, we integrate DALICC (Data Licenses Clearance Center) (Pellegrini et al., 2019), a licensing framework, which enables additional options (or legal bases) for data sharing, (iii) making improvements in the contract signing process through the use of digital signatures and providing an option to verify the signed contract for integrity, and (iv) introducing SHACL repairs for CCV data inconsistencies. In loosely-coupled contracting applications, it can happen to have multiple messages or states received, which leads to violations of the CCV consistency requirements. For such cases, we provide a robust, fault-tolerant architecture, which employs repairs to self-correct identified violations and enforce CCV consistency on the data level. The use of SHACL provides the following benefits: (i) it helps to avail the benefits of semantic technology, such as KGs' connectivity, (ii) it makes validation processes easily extendable, and (iii) it supports integration (Corman et al., 2018).

Based on motivations and discussions, we form our main research question (**RQ**) as "*How can multi-legal-base GDPR-compliant data sharing be enabled by employing consent, contracts, and licenses with SHACL?*". It is comprised of the following sub-questions: (**SQ1**) *how can data validation be performed with a W3C recommendation constraint language SHACL to make it more semantically compliant?*, (**SQ2**) *how can licenses be used as a data sharing option (as a legal basis) in digital contracting services?*, and (**SQ3**) *how can we leverage SHACL repairs for constraint violations to automatically (re-)establish consistency for CCV data.* Further, other functional improvements in the CCV tool, such as the implementation of the digital signatures, and the performance evaluation of the tool and testing the correctness of SHACL validation results are also major concerns for this study.

The rest of the paper is structured as follows. Related work is described in Section 4.2, followed by preliminaries in Section 4.3. Prior discussing the approach, we first give an overview of KG and legal background in Section 4.4. In Section 6.3, we discuss the approach for enabling multiple data sharing (through consent, contract, and license), improving data validation through SHACL, enhancing digital contracting service through digital signatures, and making digital assets (e.g., contract/consent compliance tool) licensing using DALICC, introducing SHACL repairs to automatically fix CCV data inconsistencies, while the implementation and evaluation results are elaborated in Section 4.6. Further, we conclude and discuss the future work directions in Section 4.7.

[9]Pythonic refers to the use of Python code.

4.2 Related Work

In this section, we review GDPR-compliant data sharing with multiple legal bases options such as consent, contracts, and licenses. We overview GDPR-complaint data sharing solutions in terms of guidelines, compliance tools, compliance checkers, and software.

Chhetri et al. (Chhetri et al., 2022b) developed data protection by design tool for automated GDPR compliance verification based on semantically modeled informed consent that is modeled with KGs. They implemented and evaluated TOMs (technical and organizational measures), which are a key requirement according to GDPR (Art. 6). It supports only consent-based data sharing. However, in this work (Chhetri et al., 2022b), data validation is not implemented with W3C recommendations such as SHACL and does not support multiple data sharing options like license-based data sharing.

Tauqeer et al. (Tauqeer et al., 2022) developed the CCV tool to perform GDPR compliance verification checks on digital contracts. It supports not only the management of digital contracts such as B2B and Business-to-Consumer (B2C) but also scenarios where consent is not required. They implemented and evaluated TOMs, which are a key requirement according to GDPR (Art. 25 (1)). Also here (Tauqeer et al., 2022), data validation does not follow a W3C recommendation such as SHACL and does not support multiple data sharing options like license-based data sharing.

Wang and Guan (Wang & Guan, 2023) proposed a blockchain-based traceable and secure data sharing scheme. Two major objectives have been achieved by their study: (1) to design an attribute encryption-based method to protect data and enable fine-grained shared access, (2) to develop a secure data storage scheme that combines on-chain and off-chain collaboration. They also designed a smart contract-based log-tracking mechanism to store data sharing records on the blockchain and display them graphically. The proposed scheme is evaluated for performance and security. However, there is no information about contract types like B2C, and B2B and multiple data sharing options like license-based data sharing.

Barati et al. (Barati et al., 2023) proposed a GDPR-supported model in a cloud composite service for data access and transfer requests using timed transaction systems. They presented a cloud pharmacy scenario to show the connectivity of the providers in the composite service and the flow of their requests for the collection and transfer of patient data. Further, they implemented a cloud container virtualization based on the verified proposed model realizing GDPR requirements. The proposed solution only supports consent-based data sharing and does not have multiple data sharing options.

A prototype for a contract compliance checker that is only applicable to B2B interactions is shown by Starno et al. (Strano et al., 2009) in their article. The design and implementation of a Contract Compliance Checker (CCC) monitoring service, which can observe and log B2B interaction events and ascertain whether a business partner is adhering to contracts while providing the current contract specification, are described. For the CCC, they developed the contract specification language EROP, which stands for Events, Rights, Obligations, and Prohibitions. This model only handles B2B transactions, is not GDPR compliant, and does support multiple data sharing options like license-based.

In light of GDPR compliance, Pantlin et al. (Pantlin et al., 2018) explain the focus on rising market practices in supplier contracts. The authors talk about how firms' supply chains are becoming more complicated as a result of greater outsourcing to the cloud or other external service providers. Additionally, difficulties with supplier contracts are explored, including audits of rights, security precautions, and sub-processors. The work does not provide any solutions or recommendations about how to comply with GDPR on B2B contracts and multiple data sharing options in contracting.

A GDPR compliance solution presented by Barati and Rana (Barati & Rana, 2022) helps cloud-based services delivery and supports cloud service providers. By generating legal questions that are stored on the blockchain for auditing purposes, they introduce the encoding scheme for GDPR requirements. They implement GDPR compliance checking using smart contracts in a blockchain test network to examine the execution cost of the process. The proposed tool does not comply with B2B contracts and does not have multiple data sharing options like license-based and consent-based data sharing.

The effect of GDPR on third-party contracts is examined by Gangl (Matthias, 2019). The author's survey can be utilized to analyze contracting parties in the area of cloud service providers in more detail. The survey's findings assess the GDPR's aim to see if GDPR helps bilateral cooperation, such as facilitated by data processing agreements among cloud service providers, in new and disruptive technologies. The author also evaluated whether blockchain technology can be a viable substitute for GDPR compliance verification. The author contends that blockchain technology might be a good substitute but with some limitations. One, in particular, cannot place everything in blockchain, not digital signatures, for example.

Guidelines for GDPR compliance verification are described by Doe (Doe, 2018) from the viewpoint of the law firm sectors. The author gives a thorough overview of the rules and requirements for law firms to comply with GDPR. The author lays out a set of criteria for the contract documentation, security requirements, training requirements, and data processing records. Regulations for confirming GDPR compliance are discussed, but no details regarding the implementations, or data sharing options like license-based, and data validations standards like SHACL are available.

Ferrari (Ferrari, 2018) discusses privacy concerns with blockchain technology. The author has looked at many features of blockchain technology that either complemented or contradicted the GDPR. For example, GDPR is designed to fit the centralized data storage paradigm. Data saved on blockchains, on the other hand, does not go out of its application, i.e., a third-party can not have control over data stored on the blockchain ledger. The author underlines that GDPR requirements (such as the right to be forgotten, data minimization, and conditions for data transit to third countries) impose additional technical requirements on the structure of blockchain-based solutions.

4.3 Preliminaries

We introduce the Shapes Constraint Language SHACL, which is the standard for constraint validation of Semantic Web, and describe recent work for repairing SHACL constraint violations on which this application is based.

4.3.1 SHACL - Shapes Constraint Language

The Shapes Constraint Language SHACL (SHACL Working Group, 2017) is the W3C's approach to bring constraint validation to the Semantic Web. Originally, the Semantic Web was designed with the idea of an open world and reasoning approaches used monotonic inference to add facts based on ontological descriptions. However, with the adoption of KGs for data integration purposes, e.g. to enable data silo consolidation in enterprises, there was a rising demand for checking data consistency and quality by applying constraints to KG. SHACL, as a W3C Recommendation and the closely related ShEx were introduced (Gayo et al., 2017) to provide constraint validation for RDF based on shapes. The main advantages of SHACL are the formalisation of constraints, grouping them into shapes, and the standardised validation report, represented in RDF form, which makes it easy to evaluate and automatically process the violations. These advantages provide a significant improvement when integrating SHACL validation into information architectures, like the CCV, in terms of stability and maintainability of software applications. SHACL validation provides consistency on the data level using a standardised approach, which provides a declarative way to ensure consistency for all participants in the information architecture independent of the implementation.

Target declarations

SHACL shapes use target declarations to apply shape constraints to nodes of a data graph. There are several options for targeting, namely listing nodes explicitly, class-based, property-based (subject or object of a property atom) or implicit targets if a shape is also declared as a class (SHACL Working Group, 2017). For example, class-based targets automatically target all nodes of a data graph that are in the extension of that class.

SHACL Core and SHACL-SPARQL

The W3C Recommendation on SHACL groups the constraints into two parts. The first part, SHACL Core, provides constraint components as a high-level vocabulary for common use cases to describe shapes (SHACL Working Group, 2017). These components represent often used constraints that can be used to define shapes using the SHACL vocabulary. The second part, SHACL-SPARQL, provides a constraint component that acts as an open interface to the SPARQL query language. Using SHACL-SPARQL, we can define custom constraints based on a SPARQL SELECT or ASK query. The query expresses restrictions on the graph data and thereby returns violations as a query result. SHACL-SPARQL provides a powerful way to go beyond SHACL Core when the Core constraint components are not sufficient to express the constraints. However, there is a drawback to this approach which we discuss in the context of this publication. When SHACL Core is not sufficiently expressive to define target nodes for a shape, we have to use SHACL-SPARQL and use the query for selecting the targets. However, targeting nodes of the graph within a SHACL-SPARQL constraint also requires the constraints to be done within the same SPARQL query, because SHACL-SPARQL constraints cannot be used to pre-filter before applying SHACL Core constraints within the shape. This mix of target selection and constraint validation is hard to maintain and error-prone, because it is not necessarily obvious which part of the query is done for targeting and which part is representing the constraint.

4.3.2 SHACL Repairs

Our recent publications introduced a novel approach to repair SHACL constraint violations by modifying the data to conform to the defined constraints (Ahmetaj et al., 2022). The SHACL repair approach is implemented as a repair program that can deduce repairs as minimal data modifications. It will generate consistent data regarding the shape constraints when applied to a data graph. The repair program implementation is available at GitHub[10]. The repairs come in minimal sets of additions and deletions (Ahmetaj et al., 2021) of triples that are determined by the repair program for an input data graph and a set of SHACL shapes to validate against. Adding the additions to the data and removing the deletions from the data will result in a new data graph that is consistent with the shape constraints and will be conformant when validated by a SHACL processor. The implementation of the repair program is done by generating a logic program that consists of rules and facts. This program represents an answer set (see Answer Set Programming (Eiter et al., 2009)) that can be processed by the Clingo (Gebser et al., 2019) solver. Answer set programming (ASP) provides minimal models as solutions to a given program. By building on this minimality of ASP models, the repair program provides minimal repairs as part of these minimal models. However, it is possible that there are multiple minimal repairs returned by a given program. For this case, the repair program uses optimization by prioritizing certain repairs over others. This is done by specifying weights for specific data elements and then minimizing or maximizing the sum of these weights.

Consider the following example.

```
:ContractShape a sh:NodeShape;
```

[10]https://github.com/robert-david/shacl-repairs

```
  sh:targetClass fibo-fnd-agr-ctr:Contract;
  sh:property [
    sh:path :hasContractStatus;
    sh:maxCount 1; ] .
```

The *ContractShape* defines a constraint for a maximum of one triple (atom) of the property *hasContractStatus*. Consider the following data graph.

```
:contb2b_bfcff2dc a fibo-fnd-agr-ctr:Contract;
  :hasContractStatus :statusPending, :statusFulfilled.
```

The SHACL repair program will propose two different minimal models to repair the data graph to satisfy the shape constraint. These models contain the deletion sets D_1 and D_2 respectively.

$$D_1 = \{hasContractStatus(contb2b\_bfcff2dc, statusPending)\}$$
$$D_2 = \{hasContractStatus(contb2b\_bfcff2dc, statusFulfilled)\}$$

Picking one of these sets and applying it to the data graph for deletion will result in a data graph which conforms to the maximum cardinality constraint defined in the shape. However, the choice for one of these two repair models is not made by the repair program and has to be done by the user. To summarise, the SHACL repair program provides a way to fix a data graph with minimal changes to conform to given SHACL constraints.

4.4 KG Overview and Legal Background

This section summarizes the KG overview and GDPR legal background for data sharing. As mentioned earlier, besides the contract, we are extending our previous work (Tauqeer et al., 2022) by adding another GDPR legal basis in the form of a license, which gives contractual parties more control over data. Consent legal basis was integrated into our previous work (Tauqeer et al., 2022) by the integration of “Data Protection by Design Tool for Automated GDPR Compliance Verification Based on Semantically Modeled Informed Consent” (Chhetri et al., 2022b). A major challenge faced by organizations, when complying with GDPR, is the adoption of “principles of data protection by design” and “data protection by default” (Rec. 78). These principles require the implementation of technical and organizational measures (TOMs) that ensure data integrity and confidentiality (Art. 32 (1) and Rec. 46). Further, it ensures the prevention of unauthorized disclosure or access to personal data (Art. 32 (2)). All these things add another layer of complexity. Translating these legal requirements into code is a challenging task. In both works (Chhetri et al., 2022b; Tauqeer et al., 2022), we have achieved this by the Standard Data Protection Model (SDM)[11].

[11]dataprotectionmodel

4.4.1 GDPR and Relevant TOMs

Both GDPR-complaint tools (Chhetri et al., 2022b; Tauqeer et al., 2022) follow the "data protection by design and by default" principle to ensure compliance verifications based on consent and contracts. A major contribution was the implementation of "appropriate technical and organizational measures which are designed to implement data protection principles [. . .] in an effective manner and to integrate the necessary safeguards into the processing in order to [. . .] protect the rights of data subjects" (Art. 25 (1)). The adoption of internal policies (Rec. 78) was also included in this implementation. With this, DC or DP is responsible for the implementation of TOMs in order to ensure and demonstrate that processing is carried out in accordance with the GDPR (Art. 4 (7), Art. 24). Another benefit of the implementation of TOMs is to mitigate the risk to natural persons' rights and freedoms presented by the processing of their personal data (Art. 32 (1)). The SDM offers adequate mechanisms for translating GDPR legal bases into TOMs (Chhetri et al., 2022b). A summary of GDPR requirements mapped with their data protection goals is described in ("SDM", 2020) (Table in Section C2, p. 28). The SDM is as follows:

1. Systematises data protection requirements in the form of protection goals;
2. Systematically derives generic measures from protection goals, supplemented by a catalog of reference measures;
3. Systematises the identification of risks in order to determine protection requirements of the data subjects resulting from the processing; and
4. Offers a procedure model for modeling, implementation, and continuous control and testing of processing activities.

Table 4.1 summarized GDPR principles and associated SDM protection goal, TOMs, and TOMs implementation in the tools developed in (Chhetri et al., 2022b; Tauqeer et al., 2022).

4.4.2 Legal KG

A KG, which is based on the smashHitCore (Kurteva et al., 2023) ontology and is maintained in GraphDB, is the primary data source for managing digital assets licensing, performing consent and contracts compliance verifications, and enabling license as another GDPR legal basis for data sharing. The KG modeled informed consent in accordance with GDPR Art. 7 and Rec. 32, while the contract is modeled in accordance with GDPR Art. 24, 28, and 32. The other concepts such as sensor data, data processing, license, and the involved entities are represented as well. For informed consent according to (Chhetri et al., 2022b), the KG represents its (1) state (e.g., granted/not granted, revoked, withdrawn), (2) purpose, (3) duration, (4) the requested data and its

Table 4.1: A summary of GDPR principles and associated SDM protection goals, and relevant TOMs from the GDPR-compliant tools (Chhetri et al., 2022b; Tauqeer et al., 2022).

GDPR Principles	**SDM Protection Goal**	**TOM**
Purpose limitation (Art. 5(1)(b))	Unlinkability	Role concepts with graduated access rights on the basis of identity management and secure authentication process.
Storage limitation (Art. 5(1)(e))	Availability	Documentation of data syntax.
Lawfulness, fairness and transparency (Art. 5(1)(a))	Transparency	Documentation of consents, their revocations and objections.
Accuracy (5(1)(d)), Integrity and confidentiality (Art. 5(1)(f))	Intervenability	Operational possibility of compiling, consistently rectifying, blocking and erasure of all stored personal data.
Integrity and confidentiality (Art. 5(1)(f))	Confidentiality	Encryption of data.
Integrity and confidentiality (Art. 5(1)(f)), Accountability (Art.5(2))	Integrity	Protection against external influences.
Data minimization (Art. 5(1)(c))	Data minimization	Reduction of non-required attributes of data subjects.

type (e.g., sensor data), (5) types of data processing (e.g., analysis, retrieval, adaptation, collection) associated with consent, (6) entities related to consent (e.g., DC (or DP), data subject, etc.), and (7) the time and date of a consent state change. On the other hand for a data sharing contract according to (Tauqeer et al., 2022), the KG represented its status (e.g., created, updated, violated, pending), (2) purpose, (3) contract category (e.g., business to business, business to consumer), (4) contract type (e.g., written, oral), (5) list of contractors, (6) list of contractual terms, (7), list of contractors' signatures, (8) medium (e.g., online), (9) consent id (uses to store a consent reference), (10), license id (uses to store a license reference), (11) effective date, (12) execution date, (13) end date, (14) a data controller/data processor, and (15) data subject. In addition, for digital assets licensing, the KG represents its (1) create date, (2) software description, (3) license id (uses to hold the license id from the DALICC framework), (4) asset name, and (5) version information.

Consent is obtained through consent forms that have been manually examined by legal professionals collaborating in the smashHit project against GDPR's standards for informed consent (Art.7, 12, 13, and Rec. 32). While a contract is created between DS and DC/DP through contract forms that also have been manually examined by legal professionals against GDPR's standards for a contract (Art. 24, 28, and Art. 32). Same procedure is followed for a license based contract. Data from consent or contract forms is sent over APIs to specified Simple Protocol and Resource Description Framework Query Language (SPARQL)[12] queries, which annotate and produce unique consent/contract instances in the KG. The KG can be accessed directly via the SPARQL API[13] offered by GraphDB.

4.5 Approach

This section details the conceptualization of the approach to including multiple legal bases, particularly, introducing licensing as a data sharing option (legal basis comply with GDPR), enhancing the data quality through SHACL validation, improving and securing the contract lifecycle management with digital signatures, and introducing repair strategies for SHACL repairs. First, we provide an overview of the approach (see Figure 4.1). Second, in Section 4.5.1 we explain data validation through SHACL. Thereafter, the automated clearance of rights in the form of software licenses on the CCV tool is presented in Section 4.5.2. In Section 4.5.3, enhancement in the contract signing process through digital signature is discussed. In the last, we present SHACL repair strategies in Section 4.5.4.

Personally identifiable data with a variety of data sharing options, including consent, contracts, and licenses, is the input to the CCV tool. Examples of personal information include a person's name, last name, home address, email address, driver's license

[12]https://www.w3.org/TR/rdf-sparql-query/
[13]http://smashhitactool-1.sti2.at/

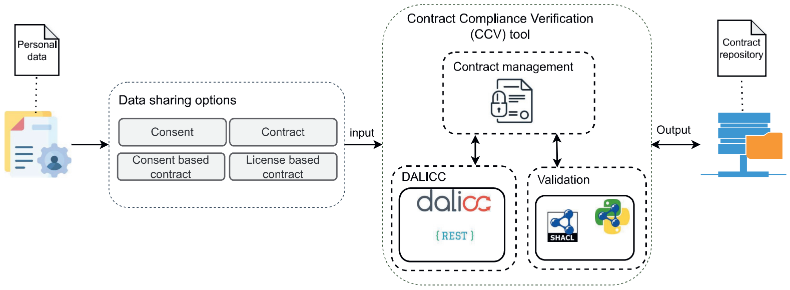


Figure 4.1: An overview of GDPR-compliant data sharing through consent, contracts, and licenses.

number, and current location. The presented here CCV tool performs the following three major functionalities: (1) implementing digital assets licensing via DALICC, (2) data validation through SHACL, and (3) enhancement in the contract signing process through digital signature. Moreover, the newly created contracts are stored in the contract repository (implemented in GraphDB[14]) which can be accessed via SPARQL Protocol and RDF Query Language (SPARQL (Pérez et al., 2009)).

The contract lifecycle management consists of six stages: contract request, negotiation, approval and signature, execution, auditing and controlling, and termination/renewal (Tauqeer et al., 2022). The most crucial ones are the auditing/controlling and termination/renewal stages, on which the CCV process is primarily focused. In our previous work (Tauqeer et al., 2022), we discussed both the contract's lifecycle management following its stages and the CCV process in detail. Further, we discuss each of the functionalities mentioned above in detail.

4.5.1 CCV Tool Data Validation Process

In the CCV tool, data validation performs on the CRUD operations (i.e., *Contract*, *Terms*, *Obligations*) and the CCV scenarios. Whenever a create/update request is made to GraphDB, the validation is performed through SHACL. Validation in the CRUD operation phase is simple where a data graph is generated based on provided data while a SHACL shape file (contains all shapes) is used for validation. On the other hand, validation is performed on compliance verification checks (i.e., CCV scenarios). Here, our primary focus is to show the validation process for the CCV scenarios. The data validation process on the CCV scenarios in the CCV tool, as depicted in Figure 4.2, comprises three parts: querying (i.e., extraction of data from GraphDB), validating retrieved information (validation), and generating validation results (i.e., validation reports).

Pandit et al. (Pandit et al., 2019c) proposed a validation model for GDPR compliance

[14]https://www.ontotext.com/products/graphdb/

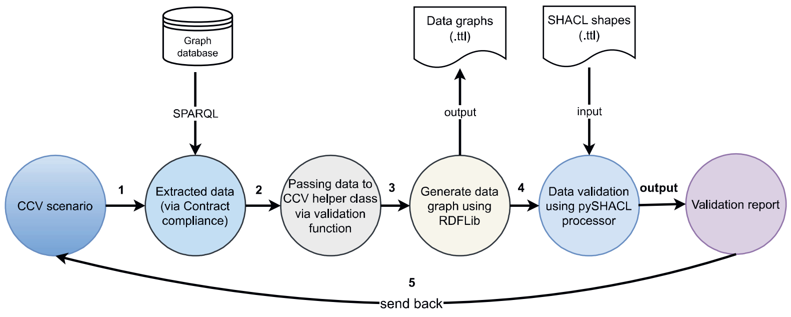

Graph database
SPARQL
Data graphs (.ttl)
output
SHACL shapes (.ttl)
input
CCV scenario
1
Extracted data (via Contract compliance)
2
Passing data to CCV helper class via validation function
3
Generate data graph using RDFLib
4
Data validation using pySHACL processor
output
Validation report
5
send back

Figure 4.2: CCV data validation process.

over provenance graphs using SHACL. The model consists of three parts: querying, validating retrieved information, and generating documentation. Since validation of retrieved information is required in our work, we follow this part. The input for the CCV module is the CCV scenarios (see details in Section 3.3 (Tauqeer et al., 2022)). After taking scenarios as the input, the CCV module extracts data from the contracts repository via class *ContractCompliance* in the first step. It then passes the extracted data to the *CCVHelper* class to generate data graphs in the second step of the validation process. The *CCVHelper* class contains a function *shacl_validation* used to generate a corresponding data graph based on the provided data. The data used to be validated is based on classes utilized by the CCV tool, including *Contract*, *Contractor*, *Term*, and *Obligation*. In the third step of the validation process, the validation function creates data graph files in turtle format, which are then passed to the pySHACL[15] processor. A file containing all the SHACL shapes, in our case, the *CCV1stScenarioShape* shape (i.e., name of a shape) is used as an input to the processor parallel in the fourth step of the data validation process. Based on the provided inputs, the pySHACL processor produces a validation report (see Section 4.6.3.2). Furthermore, these validation results (i.e., pySHACL generated validation report containing violation messages) return back to the CCV scenarios in the fifth step. In the case where the validation result conforms to “true”, no action is required from the CCV tool. But if there is a case where the validation result is “false”, it stores results locally and later shows them to the user with all violation messages.

To illustrate the CCV validation process, we elaborate on it with an example. Let us take CCV scenario B2B contracts without consent as an example, as discussed in (Tauqeer et al., 2022). In this scenario, a data controller (who wants to process data) and a data subject (who wants to share data) were involved in the form of a B2B contract. Both agreed on defined contractual terms and conditions. A data graph generated by the validation function based on this scenario is shown in Listing 4.1, while the SHACL shape used for this example is shown in Listing 4.2.

[15]https://github.com/RDFLib/pySHACL

Listing 4.1: An example of the generated data graph.

```
@prefix base: <http://ontologies.atb-bremen.de/smashHitCore#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix fibo-fnd-agr-ctr: <https://spec.edmcouncil.org/fibo/ontology/
                                                FND/Agreements/Contracts/> .

base:contb2b_bfcff2dc-2ed3-11ed-be7d-3f8589292a29 a base:CCV1st;
base:contractType base:Written;
base:forPurpose "data sharing between Tim and Alice";
base:hasContractCategory base:categoryBusinessToBusiness;
base:hasContractStatus base:statusExpired;
base:hasEndDate "2023-09-07T17:14:32"^^xsd:dateTime;
fibo-fnd-agr-ctr:hasEffectiveDate "2022-09-07T17:14:32"^^xsd:dateTime;
fibo-fnd-agr-ctr:hasExecutionDate "2022-09-07T17:14:32"^^xsd:dateTime ;
base:hasStates base:stateInvalid;
base:currentDateTime "2023-02-27T11:12:17"^^xsd:dateTime;
base:hasConsentState "empty";
base:consentId "";
base:licenseId "The MIT License" .
```

Listing 4.2: SHACL shape of the first CCV scenario.

```
@prefix base: <http://ontologies.atb-bremen.de/smashHitCore#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix schema: <http://schema.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
schema:CCV1stScenarioShape a sh:NodeShape ;

 sh:targetClass base:CCV1st;
 sh:and (
        [a sh:NodeShape;
         sh:property [
              sh:path base:hasContractCategory;
              sh:in (base:categoryBusinessToBusiness)
                ]
        ]
        [a sh:NodeShape;
         sh:property [
              sh:path base:hasConsentState;
              sh:in ("empty")
                ]
         ]
);
 sh:sparql [
 a sh:SPARQLConstraint ;
 sh:message "Violation occure the end date already passed." ;
 sh:select '''
    SELECT $this
    WHERE {
     $this base:hasStates ?state .
     $this base:hasEndDate ?edate .
     $this base:currentDateTime ?cdate .
     $this base:hasContractStatus ?cstatus .
     $this base:licenseId ?license .
    FILTER ((?state=base:statePending) && (?cdate > ?edate) && (?cstatus IN
              (base:statusCreated, base:statusUpdated, base:statusPending)))
    }''' ;
 ] .
```

The data graph comprises namespaces and data in the form of semantic triples[16]. While the shape graph contains a SPARQL-based constraint[17] (defining SPARQL-based restrictions) and two property constraints[18] (defining restrictions on the property such as contract status). Contract category and consent state are property constraints that validate the provided data graph for violations. Further, the SPARQL query filtered the records based on consent state, current date, contract status, obligation state, and license through the SPARQL constraint. The source code, data graphs, and shapes graphs used for this research are available on GitHub [19].

4.5.2 CCV Tool License Management Process

The final task in this work is to make licenses as another GDPR-compliant legal basis for data sharing and to associate automated clearance of rights on digital assets in the form of licenses. To implement licenses in the CCV tool, we integrate the DALICC framework, a software framework with semantic licenses that supports legal experts, innovation managers, and application developers in the legally secure re-utilization of third-party digital assets such as data sets, software or content (Pellegrini et al., 2019). The license management process of the CCV tool is presented in Figure 4.3.

In the first step, licenses are extracted from DALICC via REST APIs endpoints. The extracted licenses then, in the second step of the process, are associated with resources, such as datasets, software (e.g., consent and contract compliance tools), or a contract.

[16]https://en.wikipedia.org/wiki/Semantic_triple
[17]https://www.w3.org/TR/shacl/#core-components-shape
[18]https://www.w3.org/TR/shacl/#constraints
[19]https://github.com/AmarTauqeer/Contract-license

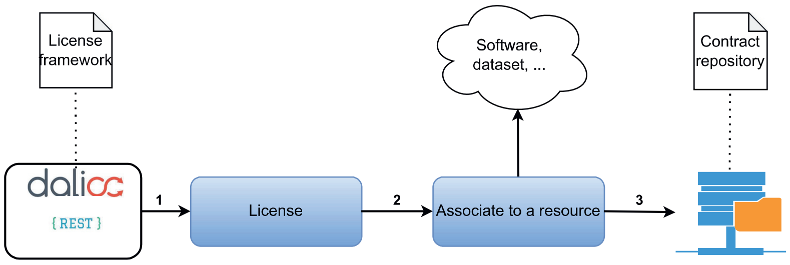


Figure 4.3: CCV license management process.

On one hand, a contract with a license becomes a license-based data sharing legal basis where contractors can define other contractual clauses not only for contracts but also for licenses. While on the other hand, the purpose of the integration of licenses is to associate rights on digital assets, such as software, source codes, and datasets. A typical example of this is to associate the MIT license (from DALICC) with the ACT tool (Chhetri et al., 2022b; Tauqeer et al., 2022). More details about the implementation of licenses can be found in Section 4.6.2

4.5.3 Improvement in the Contract Signing Process

Another major improvement in contract lifecycle management is the improvement in the approval or signing stage with security measures. The 3rd stage of the contract lifecycle management is the approval or signing (Tauqeer et al., 2022), where the contractors have to sign the contract to start its execution. Previously in (Tauqeer et al., 2022), there were only encrypted signatures in the signing process, and the tool did not comply with digital signatures (Subramanya & Yi, 2006). Protecting the contract signing process requires the contractor's digital signature verification in the contract creation stage. A contract cannot be generated without the digital signature verifications of the contractors. With digital signatures, the contract signing process of the CCV tool is secured and improved. Figure 4.4 shows the response to the creation of

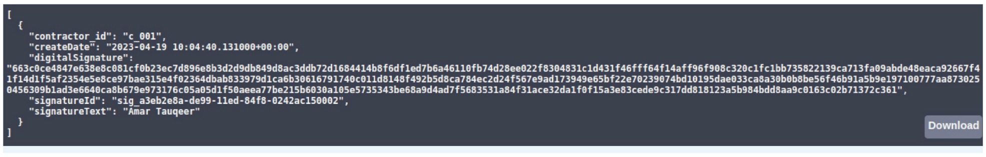

```
[
  {
    "contractor_id": "c_001",
    "createDate": "2023-04-19 10:04:40.131000+00:00",
    "digitalSignature":
"663c0ce4847e638e8c081cf0b23ec7d896e8b3d2d9db849d8ac3ddb72d1684414b8f6df1ed7b6a46110fb74d28ee022f8304831c1d431f46fff64f14aff96f908c320c1fc1bb735822139ca713fa09abde48eaca92667f4
1f14d1f5af2354e5e8ce97bae315e4f02364dbab833979d1ca6b30616791740c011d8148f492b5d8ca784ec2d24f567e9ad173949e65bf22e70239074bd10195dae033ca8a30b0b8be56f46b91a5b9e197100777aa873025
0456309b1ad3e6640ca8b679e973176c05a05d1f50aeea77be215b6030a105e5735343be68a9d4ad7f5683531a84f31ace32da1f0f15a3e83cede9c317dd818123a5b984bdd8aa9c0163c02b71372c361",
    "signatureId": "sig_a3eb2e8a-de99-11ed-84f8-0242ac150002",
    "signatureText": "Amar Tauqeer"
  }
]
```

Figure 4.4: Digital signature associated with a contractor.

the digital signature of a contractor, which verifies in the contract creation process. It contains the contractor's signature information including a digital signature. Without this digital signature, a contractor cannot be entered into a contract.

4.5.4 SHACL Repair Strategies

In this section, we introduce repair strategies to automatically ensure data consistency for the contract lifecycle. We first elaborate on formal consistency requirements (CR) of CCV and define them using logical expressions. We focus on simple CRs for the states of contract and obligation of the semantic model to explain our approach. We see these CRs also as the most basic requirements regarding robustness from the distributed systems point of view. We then introduce SHACL shapes to represent these requirements using SHACL constraints and thereby have a standardised semantic description for them. We use the SHACL repair program to determine repair models to fix the constraint violations. However, we need to automatically determine exactly one (optimal) repair for implementing a self-correcting component as part of the CCV architecture. Therefore we finally introduce repair strategies, where we formalise the necessary steps for the system to take an automatic decision.

4.5.5 CCV Consistency Requirements

The CCV consistency requirements mainly address functional requirements for the lifecycle status, where we need an unambiguous semantic description. Also, the state of obligations can affect the status of a related contract. The CCV consistency requirements pose new challenges as they require to capture the expressivity of the *logic implication* using SHACL shapes. In the following, each consistency requirement is elaborated in textual form, supported by an example and then captured using formal logic. We present two examples for CCV consistency requirements, which are selected for scope and simplicity reasons. We note that further (and more complex) consistency requirements exist as well for CCV.

CR-1. The first consistency requirement is about the functionality requirement, i.e., uniqueness, namely: *Contracts should have exactly one contract status.* We formalize these rules using the logic as below.

$$\begin{aligned}\forall x \exists y, z.\ & Contract(x)\ \wedge hasContractStatus(x, y)\\ & \wedge hasContractStatus(x, z) \wedge y \neq z \implies \bot\end{aligned}$$

CR-2. The second consistency requirement is more specifically addressing the values of the first consistency requirement: *A Contract status has to be one of pending, fulfilled or violated.* We note that contracts, as defined by the Contract Ontology (Tauqeer et al., 2022), can also initially have a contract status of created. However, when processed by CCV, this contract status is no longer accepted and the contract has to be (at least, regarding the lifecycle) in status pending. We formalize these rules using the logic as below.

$$\begin{aligned}\forall x \exists y, z.\ & Contract(x)\ \wedge\ hasContractStatus(x, y)\\ & \wedge\ hasContractStatus(x, z) \wedge (y \neq \text{pending}\ \wedge\ y \neq \text{fulfilled}\\ & \wedge\ y \neq \text{violated}) \implies \bot\end{aligned}$$

CR-3. The third consistency requirement is: *A Contract is violated if at least one associated Obligation is violated.* We formalize these rules using logic as below.

$$\begin{aligned}\forall x \exists y, z.\ & \mathit{Contract}(x) \land \mathit{Obligation}(y) \land \mathit{hasObligations}(x, y) \\ & \land \mathit{hasState}(y, z) \land z = \text{violated} \\ & \implies \mathit{hasContractStatus}(x, \text{violated})\end{aligned}$$

4.5.6 SHACL Constraints

In the following, we will define SHACL shapes which use constraint components to represent the consistency requirements CR1, CR-2 and CR-3. This translation of consistency requirements to SHACL is done using SHACL Core.

Functional Contract Status. The first shape implements the functionality and value requirements of a *Contract*, as described in CR-1 and CR-2.

```
:FunctionalContractStatusShape a sh:NodeShape;
  sh:targetClass fibo-fnd-agr-ctr:Contract;
  sh:property [
    sh:path :hasContractStatus;
    sh:in ( :statusPending :statusFulfilled
            :statusViolated );
    sh:minCount 1;
    sh:maxCount 1; ] .
```

Contract Violation. The second shape implements the dependency requirement of a contract regarding any associated obligation, where one violated obligation leads to a violated contract, as described in CR-3.

```
:ContractViolationShape a sh:NodeShape;
  sh:targetClass fibo-fnd-agr-ctr:Contract;
  sh:or (
    [ sh:not [
      sh:property [
        sh:path ( :hasObligations :hasState );
        sh:hasValue :ViolatedState ]; ] ]
    [ sh:property [
      sh:path :hasContractStatus;
      sh:hasValue :statusViolated; ] ] ).
```

4.5.7 Repairing CCV Constraints

Having SHACL shapes in place to implement the consistency requirements, the next step is to resolve inconsistencies when they occur. When using CCV as part of an open environment, where different systems work together, we can take advantage of our repair approach to ensure data consistency on the data level. For that, we implement a self-correcting system that is robust against programming errors of participating applications by repairing inconsistencies automatically. It can act as a basic architecture for a loosely coupled system in an open environment, where several participants can send updates on the RDF data level, while still ensuring the consistency of CCV data. For this implementation, we use the SHACL repair program as a basis to deduce repairs for the CCV data.

Introducing Repair Strategies

There is one missing step from deducing repairs to automatic repairs. In the case the SHACL repair program deduces multiple different options to repair the data, it will return all these possible (minimal) repairs. However, this is not sufficient for our CCV architecture, where we need an automatic approach to pick a single optimal choice so that we can apply the data changes automatically and without the need for manual intervention. Therefore, we discuss strategies for repair selection of the previously defined SHACL shapes, which then enable us to automatically choose one optimal repair in the case of multiple options and to pick new values as required.We note that this practical work did not cover defining a formal semantics for repair strategies. We focused on CR-1, CR-2 and CR-3 only and provide an implementation for exactly these.

Defining Repair Strategies

To apply repair strategies to SHACL repairs, we developed a functional prototype to expand the existing SHACL repair processor with additional rules to resolve multiple repair options. To define the repair strategies, we introduce an RDF-based vocabulary extending SHACL. This is a proposal for repair strategies only in the context of this prototypical implementation. We also explain the semantics informally:

- *sh:RepairStrategy* is a class used to state a repair strategy, which can define a set of *sh:repair* elements as advises to the repair processor how to resolve multiple repair options.
- *sh:repair* is a predicate, where the object is a blank node, which defines how to resolve a specific violating situation with multiple options.
- *sh:path* has the same semantics as defined in the W3C recommendation

and represents a path predicate.

- *sh:action* defines the kind of repair action this repair strategy is applied to. The object to this predicate can only have two options, which are *sh:add* and *sh:delete*.
- *sh:preserveOrder* defines a list of values for the value nodes of the *sh:path* predicate that represent a priority order to prioritise the different repair options.
- *sh:value* is used to represent a specific value for a value node to be added or deleted (depending on *sh:action*). This can also be negated using *sh:not* to avoid a specific value to be added or to preserve a specific value from deletion.
- *sh:AnyValue* is a placeholder for any value. Repairs are therefore applied regardless of the concrete value in the data.

We implemented a repair strategy program on top of the SHACL repair program to generate additional rules for the ASP repair program. In the following, we provide repair strategies for the examples and we the provide logical rules that extend the repair program with constraints and optimisation rules to implement these strategies.

Repairing CR-1 and CR-2.

CR-1 needs to be repaired by determining a single *hasContractStatus* property atom to keep and removing the other *hasContractStatus* atoms. The number of models equals the number of *hasContractStatus* atoms.

CR-2 states that a contract or obligation cannot be both pending, fulfilled and violated at the same time, and it needs exactly one of these values to be in the data graph. Because the states are mutually exclusive, we need to pick one depending on which are explicitly stated in the data. If a contract or clause is violated, we should keep this state and delete all other states. If a contract or clause is fulfilled, but not violated, then we should keep fulfilled. In other cases the state is pending. If there is no state, we choose to add the pending state to indicate that the contract or obligation is still open to be processed. We formalise this repair strategy, which consists of two repairs.

```
:FunctionalContractStatusStrategy
  a sh:RepairStrategy;
  sh:repair [
    sh:path :hasContractStatus;
    sh:action sh:delete;
    sh:preserveOrder (
      :statusViolated :statusFulfilled
      :statusPending ) ];
  sh:repair [
    sh:path :hasContractStatus;
    sh:action sh:add;
    sh:value :statusPending ] .
```

The repair strategy contains two repairs. The first repair for deletions will determine a single model based on weights from the 3 possible repair models (one option for each of the 3 possible values). We define a preference order using *sh:preserveOrder* with decreasing weights to implement it. The option for *statusViolated* gets the highest weight, followed by *statusFulfilled* and finally *statusPending*. The weights enable to repair program to make a decision for the optimal choice, which was not possible without this repair strategy. The implementation of the repair strategy is done by adding the following ASP rules to the repair program:

$$\#minimize 1@0, X : del(hasContractStatus(X, statusPending)).$$
$$\#minimize 2@0, X : del(hasContractStatus(X, statusFulfilled)).$$
$$\#minimize 3@0, X : del(hasContractStatus(X, statusViolated)).$$

These optimisation rules implement the repair for deletions by prioritising the values as a preference order. They apply the weights to the possible repair models, thereby picking them for deletion in increasing order of the weight (first 1, then 2, then 3). *statusPending* receives the minimum weight, meaning it will be picked as the preferred option for deletion. Second is the *statusFulfilled*, which is picked when there is no *statusPending*. Last, there is the *statusViolated*, which is only preferred to be deleted if there are no other options. In our scenario, this means that *statusViolated* will always be kept, because of the *sh:minCount* 1 requirement. Using this semantics, we can ensure that *statusViolated* is never removed, while *statusFulfilled* is only removed if there is a *statusViolated*.

The second repair for additions will determine a preferred option to be added when there is not yet a value from *statusViolated*, *statusFulfilled* or *statusPending* in the data graph. Similar to deletions, we define a preference order. In this case we maximise picking the value *statusPending*, which is the preferred value, by assigning a weight of 1, while other values do not get a weight assigned. We add the following optimization rule to the repair program:

$$\#maximize 1@0, X : add(hasContractStatus(X, statusPending)).$$

This optimization rule will maximize picking the addition of the *statusPending*, meaning it will add it as a preferred option over other values. We will illustrate the repair strategy with an example.

```
:contb2b_bfcff2dc a fibo-fnd-agr-ctr:Contract;
  :hasContractStatus :statusViolated, :statusPending,
  :statusFulfilled .
```

This data graph has 3 different values for *hasContractStatus*, which clearly violates the *FunctionalContractStatusShape* and we get the 3 repair models with deletion sets D_1, D_2 and D_3.

$$\begin{aligned}
D_1 = \{ &: hasContractStatus(: contb2b\_bfcff2dc, : statusViolated),\\
&: hasContractStatus(: contb2b\_bfcff2dc, : statusPending)\}\\
D_2 = \{ &: hasContractStatus(: contb2b\_bfcff2dc, : statusPending),\\
&: hasContractStatus(: contb2b\_bfcff2dc, : statusFulfilled)\}\\
D_3 = \{ &: hasContractStatus(: contb2b\_bfcff2dc, : statusViolated),\\
&: hasContractStatus(: contb2b\_bfcff2dc, : statusFulfilled)\}
\end{aligned}$$

When we apply the repair strategy to it, the application will prioritise the models based on the weights for the different values, resulting in a single optimal model to be picked:

$$\begin{aligned}
D_2 = \{ &: hasContractStatus(: contb2b\_bfcff2dc, : statusPending),\\
&: hasContractStatus(: contb2b\_bfcff2dc, : statusFulfilled)\}
\end{aligned}$$

D_2 will be picked as the model with the lowest weight. D_1 weights for $4(3+1)$, D_2 weights for $3(1+2)$ and D_3 weights for $5(3+2)$. Clearly, D_2 has the lowest weight of the 3 models and is therefore picked by the optimisation rules, which minimise the weights, as the single optimal repair.

Repairing CR-3.

CR-3 states that a contract is violated if at least one obligation is violated. In terms of repair strategy this means that there must not be any non-violated contract if there is a violated obligation. In this case, the repair program either removes the *ViolatedState* from the obligations or it adds the *statusViolated* to the contract. The repair strategy we choose to implement the CCV semantics is to only allow adding *statusViolated* for *hasContractStatus*. We do not want the obligations to be fixed when they have *ViolatedState* for *hasState* as this would not represent the actual real-world situation.

```
:ViolatedContractStrategy
  a sh:RepairStrategy;
  sh:repair [
    sh:path :hasObligations;
    sh:action sh:delete;
    sh:not [ sh:value [ a sh:AnyValue; ];];];
  sh:repair [
    sh:path :hasState;
    sh:action sh:delete;
    sh:not [ sh:value :ViolatedState; ];] .
```

The following constraints are added to the repair program:

$$:-del(hasObligations(X, \_)).$$
$$:-del(hasState(X, ViolatedState)).$$

The repair strategy prevents picking the deletion of *ViolatedState* as a repair choice. Furthermore, it prevents the deletion of any *hasObligations* properties (represented by _ in the constraint), which means disconnecting the contract from the obligation is no longer a valid repair choice. The alternative repair choice will then be the addition of *statusViolated* as a value for *hasContractStatus*. Again, we will illustrate the repair strategy with an example.

```
:contb2b_bfcff2dc a fibo-fnd-agr-ctr:Contract;
  :hasContractStatus :statusFulfilled;
  :hasObligations :ob_9e2bb1ce .
:ob_9e2bb1ce a :Obligation;
  :hasState :ViolatedState .
```

Without the repair strategy, the repair program would return the following single minimal model with the deletion D_1:

$$D_1 = \{hasState(ob\_9e2bb1ce, ViolatedState)\}$$

This minimal repair would violate the CCV semantics. However, when we apply the repair strategy, the repair program will prevent models to pick *ViolatedState* to be deleted, thereby preventing the single minimal model to be picked. The consequence is a new minimal model under the constraint added by the repair strategy, which has additions A_1 and deletions D_1:

$$A_1 = \{hasContractStatus(contb2b\_bfcff2dc, statusViolated)\}$$
$$D_1 = \{hasContractStatus(contb2b\_bfcff2dc, statusFulfilled)\}$$

With repair strategies, the data can now only be repaired when assigning the contract status *statusViolated* to the contract, thereby implementing the expected semantics regarding data consistency.

4.6 Implementation and Evaluation

In this section, we describe the implementation details of the CCV tool based on use cases (see details in Section 4.1) and industrial requirements. The information regarding TOMs, SDM protection goals, and the GDPR articles used in this study are presented in Table 4.1. However, more detailed information about the aforementioned resources can be found in our previous works (Chhetri et al., 2022b; Tauqeer et al., 2022). Here, we present the performance evaluation (in terms of execution time) after implementing data validation through SHACL and licenses on digital assets, implementing SHACL repairs, and the evaluation of SHACL results. We give an overview of the system setup used for this experiment in Section 4.6.1. The implementation details are described in Section 4.6.2, while the evaluation findings are reported in Section 4.6.3.

4.6.1 System Setup for Evaluation

The libraries and software that were used in this implementation are shown in Table 4.2. These libraries and software are selected due to our tool's requirements. As an example, due to having capabilities such as more intuitive data visualization, storage, and management, GraphDB was selected. Further, A Linux distribution with characteristics: (1) a system with 32 GB (gigabyte) random access memory (RAM), (2) a 1.7 gigahertz (GHz) AMD Ryzen 7 PRO 4750U processor, and (3) 1 terabyte (TB) storage, is used for deploying the service layer.

Table 4.2: List of software (or libraries) that were used in the implementation.

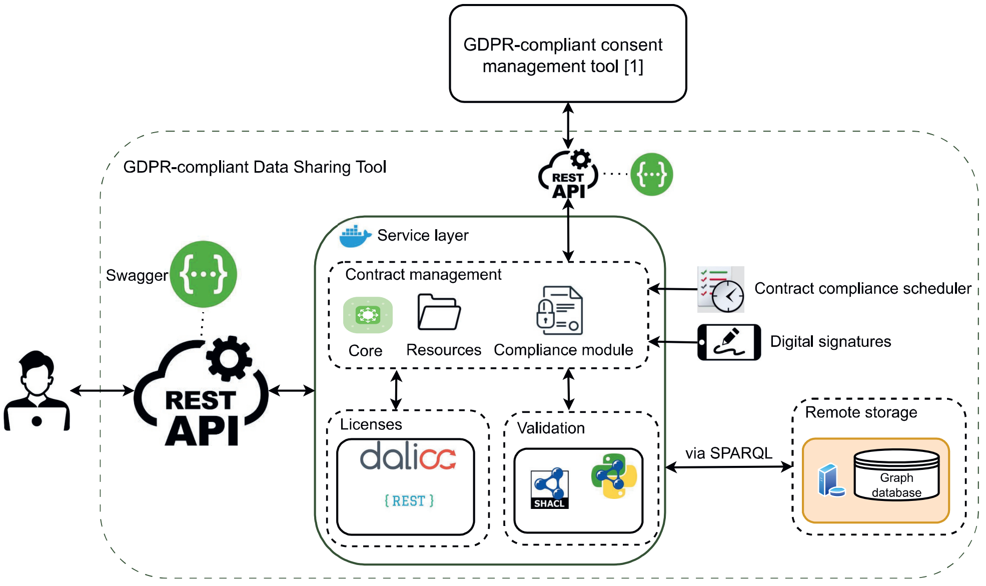
Software/Libraries	**Version**
Python ("Python", 2021)	3.9
Flask ("Flask", 2021)	1.1.2
Flask-RESTful ("Flask-RESTful", 2021)	0.3.8
Flask-SQLAlchemy [20]	2.5.1
Python Requests	2.25.1
Flask Apispec ("Flask-Apispec", 2018)	0.11.0
Pycryptodome ("PyCryptodome", 2014)	3.10.1
SPARQLWrapper ("SPARQLWrapper", 2008)	1.8.5
Docker (Community Edition) ("Docker", 2013)	20.X
SQLite	2.6
GraphDB free edition (Sirma Group, 2019)	9.4.1
Protégé	5.5.0
Pyjwt ("PyJWT", 2015)	1.7.1

4.6.2 Implementation

The CCV tool architectural design is extended and improved with data validation through SHACL (for improving data quality), by implementing licenses on digital assets such as software, implementing SHACL repairs, and with the implementation of digital signatures (for securing the contract signing process). In (Tauqeer et al., 2022), we have described each component of the CCV tool and performed data validation using ad-hoc solutions. The extended version of CCV tool is depicted in Figure 4.5. Three significant differences between the previous CCV tool architectural design and this version are: (1) a data validation module is implemented between the service layer and GraphDB. The validation is performed on data whenever a SPARQL query executes, (2) separation of the CCV module from other resources and extended its functionality with classes *CCVHelper* and *License*, (3) implementing automated clearance of rights on digital assets (such as data, software) re-using DALICC. In order to make data in a semantic-compliant manner, we implemented data validation through SHACL. To perform validation through SHACL, a data graph and a shape graph are required (see Listing 4.2 and Listing 4.1). The validation data (i.e., used to be validated) is utilized to generate a data graph through pySHACL processor. A *shacl_validation* function in the CCV module is implemented to produce a data graph using RDFLib[21], as shown in Listing 4.1 (further information is available in Section 4.5.1). Further, Listing 4.2 presents a SHACL shapes graph. In order to make data in a semantic-compliant manner, we implemented data validation through SHACL. To perform validation through SHACL, a data graph and a shape graph are required (see Listing 4.2 and Listing 4.1). The validation data (i.e., used to be validated) is utilized to generate a data graph through pySHACL processor. A *shacl_validation* function in the CCV module is implemented to produce a data graph using RDFLib[22], as shown in Listing 4.1 (further information is available in Section 4.5.1). Further, Listing 4.2 presents a SHACL shapes graph.

Another significant feature is introducing a licensing module comprised of two major classes *License* and *SoftwareLicense*. The former is used to make a contract as a license-based GDPR-compliant data sharing legal basis. This can be achieved by providing *licenseID* to a contract (a new field is added to the contract schema) in the creation process. While the latter is used to associate licensing with digital assets, e.g., datasets, consent and contract compliance tools. *License* class has two sub-classes: *DaliccLicense* and *DaliccLicenseById*. The former is used to retrieve all the licenses from the DALICC library, while the latter is used to extract a specific license based on the license id. Sim-

[21]https://rdflib.readthedocs.io/en/stable/
[22]https://rdflib.readthedocs.io/en/stable/



[1] Chhetri, Tek Raj, et al. "Data Protection by Design Tool for Automated GDPR Compliance Verification Based on Semantically Modeled Informed Consent." Sensors 22.7 (2022): 2763.

Figure 4.5: An overview of the extended CCV architecture along with license, SHACL validation, and digital signatures.

ilarly, the REST API endpoints are created for each to interact with the CCV tool. In addition, a class called *DigitalSignature* is implemented in the security module for protecting and verifying digital contracts. To achieve this purpose, two functions: *digital_signature* and *digital_signature_verify* were implemented. The former is used to create digital signatures, while the latter is used to verify digital signatures. We implemented a repair strategy program on top of the SHACL repair program to generate additional rules for the ASP repair program. The CCV repair strategy implementation is an extension to the SHACL repair program which generates the described additional rules for the ASP repair program. These rules modify the repair program to automatically pick a single optimal choice. More information can be read on GitHub [23] and [24] about data graphs, SHACL shapes, SHACL repair strategies, source code, digital signatures, and material, which are used in this research.

[23] https://github.com/AmarTauqeer/Contract-license
[24] https://github.com/robert-david/shacl-repairs/tree/ccv-repair-strategies

4.6.3 Evaluation

This section presents the evaluation of the CCV tool (in terms of execution time), focusing on performing compliance functionalities such as contract creation, contract audit, and CCV verification checks along with a GDPR-compliant legal basis license. The CCV performance evaluation is described in Section 4.6.3.1, while the SHACL validation results based on the CCV scenarios are presented in Section 4.6.3.2.

4.6.3.1 Performance Evaluation of the CCV Tool

This section details the performance evaluation (in terms of execution time) of GDPR-compliant legal bases (i.e., license) and digital assets licensing. Specifically, in our previous work (Tauqeer et al., 2022), we had presented the performance evaluation of a contract and consent based data sharing contract, while here we only focus on the license based performance evaluation, which is presented here. Since the legal bases: consent, and licenses are part of a data sharing contract, their evaluations are the same as the evaluation of a contract, the only difference is the consent id and license id in the experiments while creating a contract.

The performance evaluation of the CCV tool is based on resources, such as contract creation, contract audit, and digital assets licensing. We performed 10 different experiments on each. One experiment on a license based contract creation/audit resource involves the completion of the following five parts: (1) the contract's basic info, (2) the contractor's info, (3) the contract's terms, (4) the contract's obligations, and (5) the contractor's signatures. The total execution time of the aforementioned five parts determines how long it takes to create/audit a license-based data sharing contract. The information regarding experiments on license based data sharing contracts (i.e., creation and audit) and digital assets licensing is provided manually for this performance evaluation and available on GitHub [25].

The results of the performance evaluation of the CCV tool are presented in Figure 4.6, 4.7, and Figure 4.8, while Table 4.3 shows a license based data sharing contract performance evaluation (in terms of the creation process).

Figure 4.6 (a) shows the performance evaluation of a license based data sharing contract (create and audit), while Figure 4.6 (b) represents a contractual term (create and audit) performance evaluation. The green bars denote license based data sharing contract creation, whereas the orange bars represent a contract audit. All the experiments are shown on the x-axis while the time spent on each

[25]https://github.com/AmarTauqeer/Contract-license/tree/master/backend/evaluation

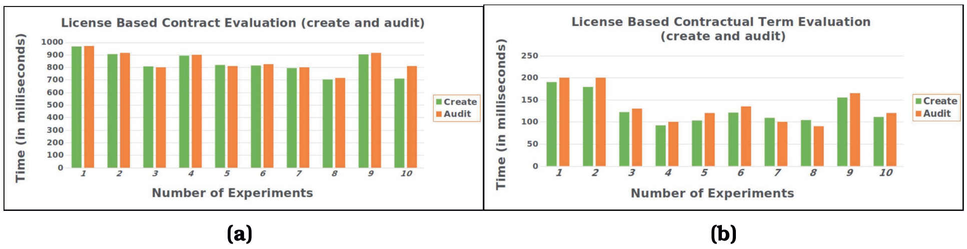


Figure 4.6: Performance evaluation of a contract and a contract term. (**a**) Time spent on contract creation and audit. (**b**) Time spent on term creation and audit.

experiment is shown on the y-axis. The time spent is measured in milliseconds on both graphs. Since this work is extended to our previous work, it used the same system setup defined in (Tauqeer et al., 2022) and in Section 4.2. In Figure 4.6 (a), the maximum time spent on a license based data sharing contract creation is 966 milliseconds (ms), and for audit is 970 ms, while the minimum time spent on license based data sharing contract creation is 703 ms and for audit is 715 ms. The average time spent on a license based data sharing contract creation is 831.4 ms whereas on a contract audit is 846 ms. Similarly, the maximum time spent on a contractual term creation is 190 ms, and for the audit is 200 ms, whereas the minimum time spent on contractual term creation is 92 ms and for the audit is 90 ms in Figure 4.6 (b). The average time spent on a contractual term creation is 128.6 ms whereas on a contractual term audit is 136 ms.

Furthermore, Figure 4.7 (a) and Figure 4.7 (b) represent the performance evaluation on contractual clauses and software licenses (i.e., digital assets licensing). The maximum and minimum times spent on both creation and audit can be seen in both figures. The average time spent on a contractual obligation creation is 182.6 ms whereas on an audit is 187.9 ms. Similarly, the average time spent on a software license creation is 328.3 ms while on an audit is 337.3 ms.

Figure 4.8 shows the performance evaluation (in terms of execution time) of the CCV module (i.e., based on a license). There were 5 different experiments performed to measure the CCV module performance evaluation. The execution time varies due to the number of license based data sharing contracts. For instance, in experiment 1 there was only one license based data sharing contract while in experiment 5, there were 3 license based data sharing contracts. This shows the execution time increases with increasing the number of contracts.

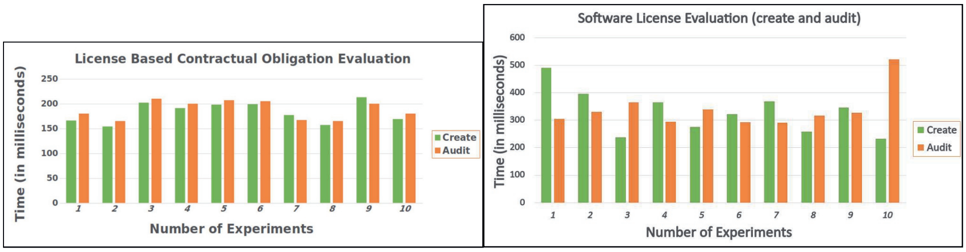


Figure 4.7: Performance evaluation on contract obligation and software license. (**a**) Time spent on obligation creation and audit. (**b**) Time spent on license creation and audit.

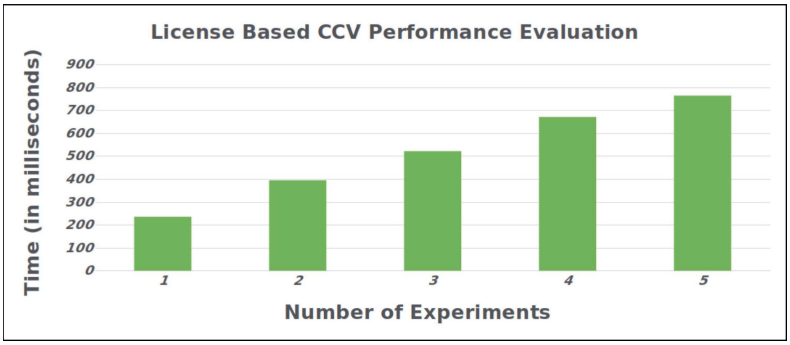


Figure 4.8: Time spent on CCV.

Table 4.3 shows a license based data sharing contract performance evaluation (only with creation experiments) based on use cases (see Section 4.1). It shows five types of time measurements, each comprised of 10 different experiments with their execution times. The minimum and maximum times spent for each type of measurement are highlighted in a bold font. Additionally, all experiments are presented with their execution time (in milliseconds) and payloads in bytes (b).

Table 4.3: An overview of contract performance evaluation in terms of execution time along with content sizes.

ID	Basic information	Contractor	Contractual term	Contractual clause	Contractor's Signature	Total
1	135 (831 b)	**325 (581 b)**	**190 (413 b)**	166 (590 b)	150 (375 b)	**966 (2790 b)**
2	183 (831 b)	246 (581 b)	179 (413 b)	**154 (600 b)**	**143 (378 b)**	905 (2803 b)
3	124 (831 b)	173 (590 b)	122 (407 b)	202 (598 b)	186 (383 b)	807 (2809 b)
4	145 (831 b)	193 (564 b)	**92 (410 b)**	191 (590 b)	**272 (382 b)**	893 (2777 b)
5	149 (831 b)	206 (574 b)	103 (419 b)	198 (600 b)	163 (374 b)	819 (2798 b)
6	113 (831 b)	176 (597 b)	121 (413 b)	199 (600 b)	205 (382 b)	814 (2823 b)
7	150 (831 b)	206 (562 b)	109 (413 b)	177 (590 b)	152 (375 b)	794 (2771 b)
8	**108 (831 b)**	190 (563 b)	104 (410 b)	157 (600 b)	144 (378 b)	**703 (2782 b)**
9	**209 (831 b)**	170 (564 b)	155 (402 b)	**213 (600 b)**	156 (376 b)	903 (2773 b)
10	114 (831 b)	**154 (573 b)**	111 (402 b)	169 (600 b)	162 (385 b)	710 (2791 b)

In summary, we draw the following main conclusions from the CCV performance evaluation: (1) the license based data sharing contract creation is a complex process having five parts. In order to create a data sharing contract these parts have to be completed. (2) It takes an average of 831 ms to complete the whole process. (3) Increasing the number of contracts also increases the execution time in CCV. and (4) The execution time depends on the payloads for each contract part.

4.6.3.2 The Evaluation of the SHACL Validation Results

There were 12 test cases derived from the CCV scenarios discussed in (Tauqeer et al., 2022) to test the correctness of the SHACL validation results. As an example, CCV's 1st scenario has been taken having a B2B contract between LexisNexis (as a data controller) and Forum Virium Helsinki (as a data processor) (see Table 4.4). Since the end date has passed, the contract violates. While performing validation through SHACL in the CCV tool, it shows the violation message "Violation: Contract end date has been passed" which proves the result is correct, as it can be seen in Table 4.4. Similarly, the remaining 11 test cases were derived from other CCV scenarios and tested. Data for all test cases were provided manually. In case of a violation, the violation messages are shown to the contractors. The violation messages depend on the violation type, such as for example, referring to the date (e.g., "Violation: Contract end date has been passed."), or the consent state (e.g., "Violation: Consent has expired already."). The information about provided data graphs, the CCV scenario, SHACL shape, the violation message, and the results are available on GitHub (Tauqeer, 2022).

4.6.3.3 The Evaluation of the CCV Repair Strategies

To verify our approach, we implemented the repair strategies on top of the SHACL repairs, defined test scenarios based on the CCV data from the previously developed architecture and performed evaluations regarding correctness and performance. First, we extended the original implementation of SHACL repairs to parse repair strategy definitions as described in this paper and produce the optimization rules and constraints to be added to the repair program. Applying the repair strategies is done on the ASP program side only, which makes it easy to integrate them with an existing repair program on the level of ASP rules. We then defined several unit tests to cover the consistency requirements defined in this paper using the provided SHACL shapes and repair strategies. For tests regarding real-world data, we used existing data from the previously developed CCV architecture (Tauqeer et al., 2022).

Simulation using synthetic Inconsistencies

The test scenario is intended to simulate inconsistencies that can arise in distributed systems which cannot guarantee a global unique status of contracts regarding the lifecycle. It is then possible for different participants in the architecture to have the same contract with different lifecycle status in the data graph. Such a situation clearly violates the consistency requirements and can be solved via the repair strategies. The approach we take to generate the data is the following. The basis is a consistent snapshot of contract and obligation data from the CCV architecture, which includes 19 contracts and 21 obligations. For these contracts and obligations, we randomly generate additional status. The randomisation is done regarding how many to add (up to 3 in addition to the existing one status) and also which ones to add (*statusPending*, *statusFulfilled*, *statusViolated*). We also randomly generate additional states for the obligations, where at least one violated obligation (*ViolatedState*) must lead to a violated contract (*statusViolated*) as well. In the following, we show an example of additional status being generated for a contract and additional states being generated for one obligation. Assume the consistent snapshot of the data graph contains the following contract and obligation data, which represents a pending contract with one pending obligation:

```
:contb2b_bfcff2dc a fibo-fnd-agr-ctr:Contract;
  :hasContractStatus :statusPending;
  :hasObligations :ob_9e2bb1ce .
:ob_9e2bb1ce a :Obligation;
  :hasState :PendingState .
```

To generate inconsistent data, we randomly pick a number of status between 0 and 4 to add to the contract, thereby adding up to 3 additional status to the already existing one. We pick a number of one status for this example and randomly pick the *statusFulfilled* to be added. We do the same for the obligation. We also pick one state to be added to the obligation and and randomly pick the *ViolatedState*. The resulting inconsistent data graph for this example is:

```
:contb2b_bfcff2dc a fibo-fnd-agr-ctr:Contract;
  :hasContractStatus :statusPending;
  :hasContractStatus :statusFulfilled;
  :hasObligations :ob_9e2bb1ce .
:ob_9e2bb1ce a :Obligation;
  :hasState :PendingState .
  :hasState :ViolatedState .
```

We can see that the contract in the example now violates CR-1, because it has more than one status, and CR-3, because it has an associated obligation with state *ViolatedState*, but it does not have a status *statusViolated*. Using this approach, we perform the simulation to evaluate the CCV repairs. We define the acceptance criteria for the simulated tests as the following

- All the constraints are repaired, resulting in a consistent data graph.
- There is only one (unique) repair model returned.

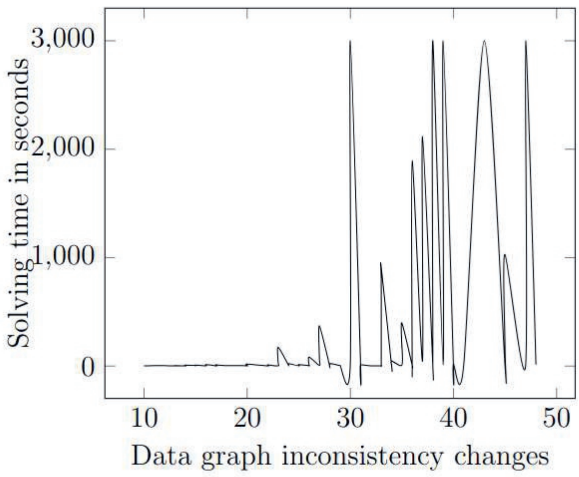
Test Cases

We performed a simulation with randomized data graphs where we generate sets of status changes for the contracts and obligations in the data graph. We generate these test sets for randomly adding up to 2, 3 and 4 picked changes for each contract and for each obligation to the data graph of the consistent snapshot. Using this strategy, we have synthetic data for a low number inconsistencies (2), a medium number of inconsistencies (3) and a high number of inconsistencies (4). For each of these options, we generate test sets containing 10 randomly generated data graphs based on the consistent snapshot. We repeat this generation 3 times for a total of 90 test cases. We then generate and perform the repair strategy program using clingo and measure the time it takes for each test case run.

Test Results

In the following, we discuss the outcomes of the CCV repair evaluation regarding correctness and performance. Of the 90 performed tests, 5 tests did not terminate regarding the clingo solving and were therefore canceled. All other 85 tests were completed with successfully with returning a single (unique) repair model, thereby satisfying the defined acceptance criteria. For discussing the performance of the CCV repairs, we generated a plot of the test performance data in relation to the inconsistencies. The plot (below) shows the relation between the time it takes to run the repair strategy program and the number of changes done to the data graph. For the non-terminating test cases, we set a time of 3000 seconds to be able to represent them in the result data and visualize them in the plot as well.

We can see that above about 30 changes to the data graph in total for contracts and obligations, which result in inconsistencies, the performance gets to a point where it was difficult to calculate a result in a realistic amount of time. This sets a limitation on the practical applicability of the approach re-



garding scalability and has to be considered regarding the scenarios where the approach is applied.

We note that the performance limitations are heavily influenced by the implementation of the preference orders. Ordering takes place as a post-processing step after the ASP program determined all models. This results in determining and ordering of a potentially large number of models, where the (unique) optimal model is then finally returned. As a conclusion based on the test results, we propose to limit the CCV architecture's processing for updates to the current data graph to < 30 global changes to contract status data. Thereby a stable performance can be achieved.

The implementation of the repair strategies, the unit tests and the two test scenarios can be viewed at GitHub[26].

4.7 Conclusion and Future Work

This study extends our previous work (Tauqeer et al., 2022) and makes the following improvements: (1) integration of the DALICC framework to associate licenses, one more legal basis fully comply with GDPR that was not present in our previous work, (2) enabling data sharing using multiple legal bases, consent, contracts, and licenses, (3) improvement of the GDPR compliance verification tasks by performing GDPR compliance verification checks on contracts

[26]https://github.com/robert-david/shacl-repairs/tree/ccv-repair-strategies

using the validation mechanism by SHACL, (4) improvement of the contracting process through the use of digital signatures that allow protection and verification of the integrity of contractual information in digital contracting, and (4) introduction of SHACL repair strategies. The proposed approach comprises multiple legal bases, such as consent, contract, consent-based contract, and license as an input, a CCV tool, and an output in the form of digital contracts, which are stored in the contract repository (implemented in GraphDB). The CCV tool consists of three components: contract management, license module, and validation module. To answer SQ1, data validation of the CCV tool is improved with SHACL (see details in Section 4.5.1) to make it semantic technology compliant. In turn, SQ2 has been answered by creating automated clearance of rights on digital assets, such as software with the integration of the DALICC framework in the CCV tool (see details in Section 4.5.2). Furthermore, we have implemented digital signatures to protect GDPR-complaint data sharing (see details in Section 4.5.3). SQ3 was addressed by providing repair strategies for an existing SHACL repair approach, which automatically fixes CCS inconsistencies in our scenario, but still has to be analyzed regarding generic applicability. We have evaluated the approach with performance tests and presented their results in Section 4.6. To ensure legal compliance and industrial relevance, our work was conducted in collaboration with legal experts and industrial partners. The outcomes are beneficial particularly to small and medium enterprises, which often lack the legal expertise and resources to access them.

The future work includes the following: (1) improving the data sharing negotiation process where the data subject will have more options to collaborate on making contract clauses, (2) automated extraction of contract clauses (e.g., from text) to make the contract creation process more efficient and fast, (3) full integration of repair strategies for SHACL constraints violations (Ahmetaj et al., 2022), and (4) automatic detection of conflicts between GDPR-complaint data sharing contractual clauses and licenses used for data sharing–it can happen when a data sharing contractual clause/obligation described same descriptions for data sharing rights as defined in standard licenses from the DALICC framework. Regarding the SHACL repair strategies approach, this work is the first step in our path to deployment to provide consistency for distributed contract sharing and consent on the data level, thereby making it less error-prone and independent of participating parties' implementations. Although the constraints presented here are rather a simple first step in this application, it is easy to expand on these and cover more complex scenarios, because the implementation is based on the standardized SHACL-Core language, which can be automatically translated for repairs. We tested the viability of our approach with a simulation using synthetic inconsistencies regarding correctness and performance. While the correctness could be verified, the performance showed limitations for practical applicability. Therefore, we look into performance improvements for the current approach or consider alternative approaches and implementations to better scale with the number of changes. A first step towards improvements is to determine how we can automatically split up the data graph into independent sets, repair each of them separately and then merge the sets back into one consistent data graph. Such pre- and post-processing steps do not require to modify the approach presented in this paper and are promising for improving scalability for the presented use case. We also aim to deploy this solution into a productive scenario with multiple distributed partners to evaluate the practical feasibility of the approach in such a practical real-world case. Furthermore, we will look into further use cases that can be covered by SHACL and investigate if the repair strategy approach is generally applicable to a wider range of repair scenarios. Finally, addressing the foundation for this approach on the theoretical side, we aim to provide a formal semantics for repair strategies.

Author Contributions: Conceptualization, A.T., T.R.C., and A.F; methodology, A.T., T.R.C., and A.F; software, A.T., and R.D.; validation, A.T., T.R.C., and A.A.; formal analysis, A.T., and T.R.C.; investigation, A.T., T.R.C., and A.F; resources, A.T., and R.D.; data curation, A.T.; writing—original draft preparation, A.T.; writing—review and editing, A.T., A.F., A.A., T.R.C., and R.D.; visualization, A.T., and R.D.; supervision, A.F.; project administration, A.F.; funding acquisition, A.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Horizon 2020 project smashHit (grant number 871477).

Data Availability: Not applicable.

Acknowledgment: We express our gratitude to Friederike Knoke and Samuel Iheanyi Nwankwo from Leibniz Universität Hannover (LUH) Institut für Rechtsinformatik (IRI) for the legal analysis of the contracts for our use cases. Further, we thank our industry collaborators LexisNexis Risk Solutions, Volkswagen AG, Infotripla and Forum Virium Helsinki for supporting the use cases for our work. We also thank our colleagues Rainer Hilscher and Antonio J. Roa-Valverde from University of Innsbruck (UIBK) for participating in the discussions and providing appropriate feedback on our work.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish

the results.

Table 4.4: An overview of scenarios, data graph, SHACL shape, violation message, and SHACL validation results. The first 3 are presented here while the rest can be seen here (List of all CCV SHACL validation results).

ID	Scenario	Data graph	Shape name	Message	Result

1	1	base:contb2b_e45dc546-2e9e-11ed-be7d-3f8589292a29 a base:CCV1st;
		base:contractType base:Written;
		base:forPurpose ”data processing agreement between LexisNexis and FVH data processor”;
		base:hasContractCategory base:categoryBusinessToBusiness;
		base:hasContractStatus base:statusUpdated;
		base:hasEndDate ”2023-04-01T10:38:07”8sd:dateTime;
		fibo-fnd-agr-ctr:hasEffectiveDate ”2022-09-07T10:38:07”8sd:dateTime;
		fibo-fnd-agr-ctr:hasExecutionDate ”2022-09-07T10:38:07”8sd:dateTime ;
		base:hasStates base:statePending;
		base:currentDateTime ”2023-04-15T11:12:17”8sd:dateTime;
		base:hasConsentState ”empty” .
			CCV1stScenarioShape	Violation: Contract end date has been passed.	Correct

2	1	base:contb2b_e45dc546-2e9e-11ed-be7d-3f8589292a29 a base:CCV1st;
		base:contractType base:Written;
		base:forPurpose ”data processing agreement between LexisNexis and FVH data processor”;
		base:hasContractCategory base:categoryBusinessToBusiness;
		base:hasContractStatus base:statusCreated;
		base:hasEndDate ”2023-09-10T10:38:07”8sd:dateTime;
		fibo-fnd-agr-ctr:hasEffectiveDate ”2022-09-07T10:38:07”8sd:dateTime;
		fibo-fnd-agr-ctr:hasExecutionDate ”2022-09-07T10:38:07”8sd:dateTime ;
		base:hasStates base:statePending;
		base:currentDateTime ”2023-04-15T11:12:17”8sd:dateTime;
		base:hasConsentState ”empty” .
			CCV1stScenarioShape	No action	Correct

3	2	base:contb2c_r6185eaa-a2e6-11ed-b1bc-0b613fb8badf a base:CCVSecond;
		base:contractType base:Written;
		base:forPurpose ”data sharing between processor and consumer”;
		base:hasContractCategory base:categoryBusinessToConsumer;
		base:hasContractStatus base:statusUpdated;
		base:hasEndDate ”2024-07-07T10:38:07”8sd:dateTime;
		fibo-fnd-agr-ctr:hasEffectiveDate ”2023-02-02T14:14:25”8sd:dateTime;
		fibo-fnd-agr-ctr:hasExecutionDate ”2023-02-02T14:14:25”8sd:dateTime ;
		base:hasStates base:statePending; base:hasConsentState ”Invalid” .
			CCVSecondScenarioShape	Violation: Consent has been expired already.	Correct

CHAPTER 5

Smell and Taste Disorders Knowledge Graph: Answering Questions Using Health Data

This chapter is based on:

Tauqeer A., Hammid I., Aghaei S., Parvin P., M. Postma A., and Fensel A. Smell and taste disorders knowledge graph: Answering questions using health data. Expert Systems with Applications, page 121049, 2023. https://doi.org/10.1016/j.eswa.2023.121049 (Tauqeer et al., 2023)

Abstract

Smell and taste disorders have become a more prominent issue due to their association with Covid-19, and their impact on quality of life and health outcomes. However, pertinent information regarding these disorders is often inaccessible and poorly organized, with the majority of data stored solely in clinical data repositories. To rectify this, a technological solution capable of digitizing, semantically modeling, and integrating health data is necessary. The knowledge graph, an emerging technology capable of organizing inconsistent and heterogeneous health data and inferring implicit knowledge, presents a viable solution to this problem. In pursuit of the aforementioned goal, an existing ontology pertaining to smell and taste disorders was enriched by introducing additional relevant concepts and relationships. Subsequently, a knowledge graph was constructed based on the defined ontology and patients' data. The resultant knowledge graph was subjected to a rigorous evaluation, encompassing dimensions such as completeness, coherency, coverage, and succinctness. The evaluation established the effectiveness and usability of the knowledge graph, with only minor issues detected through the OOPS! pitfall scanner. Furthermore, as a proof-of-concept for clinical application, a user interface was created, enabling users to access pertinent information concerning smell and taste disorders, including causative factors, medications, and etiology, among others. The interface generates a graph-based structure based on the selected question from a drop-down menu. The end-user can modify the query by merely clicking on the generated graph to ask related questions. This study showcases the potential of knowledge graphs centered on smell and taste disorders to organize and provide accessible health data to end-users.

Keywords— chemosensory dysfunction, Semantic modeling, health, data sharing, knowledge graph, ontology, question answering user interface.

5.1 Introduction

The senses of smell and taste play an essential role in our daily life. Odors are key players in food choice, social interactions, and the detection of environmental hazards (Croy et al., 2014). Many people only realize how vital the sense of smell is when they lose it. Since its appearance in 2019, the Covid-19 virus has a phenomenal impact on the world. Smell and taste disorders are among the most common symptoms of a Covid-19 infection. These changes may have a long-term impact on patients (Parma et al., 2020). However, prior to the Covid-19 pandemic, already 3 to 20% of the general population experienced smell and taste disorders (Boesveldt et al., 2017). The most common causes of these disorders are (1) head trauma, (2) viral infections, (3) nasal causes such as sinusitis or polyposis nasi, and (4) smell disorders associated with aging or neurological illnesses such as Parkinson's or Alzheimer's disease (Hummel et al., 2011). Smell and taste disorders can be associated with an increased risk of dementia, frailty, and all-cause mortality (Laudisio et al., 2019). This association underscores the increasingly recognized connection between smell and taste ability and other health domains, such as physical and cognitive function (Oleszkiewicz et al., 2020). Therefore, it is necessary to analyze data on changes in smell and taste ability from a health perspective. However, much information about smell and taste disorders, as well as their relationships to other health data, is neither well organized nor available to the general public (Kim et al., 2019). Digitizing these data and semantically modeling them is needed to make them easily accessible to end-users, such as healthcare providers and patients.

Healthcare digitization has resulted in the creation of numerous resources (e.g., software that makes health information more accessible to patients) that improve healthcare services and their efficiency and cost-effectiveness (Kuo, 2011). However, deploying various networks, such as the Internet of Things as well as publicly accessible databases and social networks, generates massive amounts of multidisciplinary health data. These data appear as information silos, and this makes it challenging to fully exploit the data. Addressing this challenge highlights the importance of applying new approaches such as ones employing *ontologies* and *knowledge graphs (KGs)*. These approaches transform information silos across the entire health system into more (re)usable data that are useful for various health-oriented applications (Kalaycı et al., 2020).

An ontology is a formal description of knowledge as (1) a set of concepts (also known as classes) within a domain (e.g., protein is a concept within the domain of molecular biology) and (2) the relationships among the concepts (e.g., an enzyme is a kind of protein, which in turn is a kind of macromolecule) (Gruninger & Lee, 2002). RDF (Resource Description Framework) schema (Pan, 2009) is a language for writing ontologies promoted by the semantic web. In RDF schema, statements and facts are represented in a triple format, including three fields: subject, predicate, and object (e.g., given the fact "drug has side effect", "drug" and "side effect" are viewed as the subject and object, respectively, and “hasSideEffect” presents their relationship as the predicate).

A diverse range of ontologies has been developed to semantically represent healthcare concepts, e.g., BioPortal Biomedical Ontology (Salvadores et al., 2012), General Medical Science (Scheuermann et al., 2009), Human Disease Ontology (Genome Sciences, Institute, 2022), Human Phenotype Ontology (Köhler et al., 2020), Gene ontology (Ashburner et al., 2000), and Upper-Level Ontology for the Biomedical Domain (McCray, 2003). Generally, ontology enhances data quality in two ways: on the one hand, by improving metadata and provenance that allows users to make better sense of the data, and on the other hand, by giving a common understanding of information and making explicit domain assumptions. As a result, the interconnectedness and interoperability of the model make it invaluable for addressing the challenges of accessing and querying data in large organizations like clinics and hospitals.

In addition, in a realistic world, new concepts are continuously developing with an incredible speed that drive ontology engineers to constantly update and enrich the generated ontologies with new concepts, terms, and lexicon, or renew a hierarchy of ideas (Petasis et al., 2011). The ontology becomes more extensive and comprehensive as a result of enrichment (Ramayanti et al., 2020). For example, the Neurological Disease Ontology is being developed as an extension of the Ontology for General Medical Science (Scheuermann et al., 2009). However, in this way, an ontology constantly requires manual efforts and resources to be enriched and maintained. Several methods have been developed in recent years to solve the issues associated with manual ontology building by automating or semi-automating the maintenance process.

The automatic ontology enrichment process uses advanced methods such as natural language processing techniques (Mellal et al., 2021) to extract knowledge from the source related to a domain of discourse. In comparison, a semi-automatic transmission combines the basics of both manual and automatic transmission (Nefzi et al., 2014). While ontology enrichment has been done in many other health subdomains, it still needs to be done for smell and taste disorders.

The ontology data model can be applied to a set of individual facts to create a knowledge graph: a collection of entities, where the types and the relationships between them are expressed by nodes and edges between these nodes (Sharma, 2022). In the abovementioned example, “drug” and “side effect”, as entities, corresponding to the nodes, and the predicate “hasSideEffect” is considered the edge. KGs are practically used to capture and organize a large amount of inconsistent and heterogeneous data that can later be explored using query mechanisms.

The KG construction approaches range from manual curation to automated techniques, such as text mining (Liu & Yang, 2022; Nicholson & Greene, 2020). To this end, data needs to be extracted and integrated from different resources ranging from unstructured data, such as plain text, to structured data, like table formats. Several studies have demonstrated the importance and application of KG in various aspects of the health domain (Aghaei et al., 2022b; Du et al., 2022; Sheng et al., 2020). However, not every aspect of the health domain has been sufficiently explored and digitized. Due to KG's expressive nature, and the corresponding ability to store and retrieve very nuanced information, Question-Answering over KG (QAKG) has received some attention from the research community. A QAKG system can be used to respond to inquiries and assist users in accessing relevant and pertinent knowledge (Aghaei, 2021; Chakraborty et al., 2019).

QAKG has received some attention in the health domain. Park et al. (Park et al., 2021) hypothesized that the graph-based question answering for Electronic Health Records (EHR) is more natural compared to tables. Zhou et al. (Zhou et al., 2021) developed a question-answering system for 14,366 different kinds of diseases. Sheng et al. (Sheng et al., 2020) also proposed a Domain Specific QA System for Smart Health Based on a Knowledge Graph (DSQA), to answer domain-specific medical questions. More recently, Du et al. (Du et al., 2022) developed a QA system to make the data related to COVID-19 available for users. Accessing the knowledge stored in KGs is facilitated through the use of formal query languages with a well-defined syntax, such as SPARQL (Swathi et al., 2016) and GraphQL (Taelman et al., 2018). However, using formal queries to access the knowledge in KGs poses difficulties for non-expert users. The users need to understand the syntax of the formal query language and the underlying structure of KG (Aghaei et al., 2022c). In contrast, a QAKG User Interface (UI) supports end-users, like patients and medical doctors, in posing natural language questions (i.e., questions asked in daily health care routines) using their terminology and receiving a concise answer (Chakraborty et al., 2019).

Our work seeks to take advantage of semantic technologies to organize and model information on smell and taste disorders, making the data more accessible to end users. With this aim, we investigated the following research questions: (1) how can available datasets and existing ontologies be used to create a KG on smell and taste disorders? and (2) how can a query be formulated and visualized, without requiring the end-user to have technical skills or knowledge of the SPARQL query language and KG?

The main contribution of this paper is the creation of a KG representing data on smell and taste disorders, as well as the development of a user-friendly query-answering interface to assist end-users in accessing the data in KG.

5.2 Material and Methods

This section outlines the specific steps and procedures that were taken in order to achieve the research objectives. The first step in the study is to identify appropriate data sources for ontology modeling and enrichment. Once the data was identified, we extracted relevant information and used this information to create a KG. The quality of the KG was then validated (see Section 5.3.6) to ensure that it was accurate and reliable. To develop the QA user interface, a variety of programming languages and software tools have been used. We designed and implemented an interface allowing users to query the KG and obtain relevant information. Overall, this section provides a comprehensive overview of the methodology used in this study.

5.2.1 Dataset

Data used in this research was collected in the Smell and Taste Center in Hospital Gelderse Vallei, Ede, the Netherlands [1]. This center is a joint clinic of Hospital Gelderse Vallei [2] and the Division of Human Nutrition and Health at Wageningen University, at Wageningen [3]. The dataset was created by manually retrieving data from the electronic files of patients who visited the center because of a smell and/or taste disorder. Patients underwent an extensive testing protocol as part of clinical care assessment, including a standardized smell test (Sniffin' Sticks (Oleszkiewicz et al., 2019)) and taste test (Taste Strips (Landis et al., 2009)). The use of clinically collected data for research purposes was approved by the local ethical committee (Review committee for scientific research of Hospital Gelderse Vallei, Ede, the Netherlands; BC/1703-143). All patients signed an informed consent form, giving permission to use their data for research.

The dataset collected in tabular format contained data from 379 patients (rows), including 66 variables for all patients (columns). The variables were divided into 7 types of information: socio-demographic factors, allergies, history of illness, assessment/diagnosis, social history, etiology, and therapy. The type of socio-demographic factors included patient age, gender, weight, height, and body mass index (BMI). The diagnostic tests used during the visit were placed under assessment. The diagnosis/assessment information includes a diagnosis (i.e., smell and/or taste disorder) based on the results of the diagnostic tests. Further, the dataset also contained information about etiology (e.g., iatrogenic, secondary, and idiopathic), treatment (e.g., smell training), allergies, and the history of illness. The history of illness included the medical conditions of the patients, complaints, symptoms, comorbidities, prescribed medications, and physical examination. The dataset employed in this study, while seemingly limited in size, was found to contain a wealth of detailed information. Specifically, the dataset provided extensive information on medication use and comorbidities, which were not previously included in the selected ontology (Clinical Smell and Taste Disorders Ontology (CSTD) [4]). As such, this dataset proved to be a valuable addition to the ontology, providing a more comprehensive and nuanced understanding of the underlying concepts.

Additionally, CSTD contains 98 classes and only 2 properties regarding smell and taste disorders. It does not have information about etiology, comorbidities, and endoscopy

[1]https://www.geldersevallei.nl/patient/afdelingen/reuk-en-smaakcentrum

[2]https://www.geldersevallei.nl/home

[3]https://www.wur.nl/en/research-results/chair-groups/agrotechnology-and-food-sciences/human-nutrition-and-health.htm

[4]https://bioportal.bioontology.org/ontologies/CSTD

and also misses much other information regarding disorders. (To add this missing information, 21 new classes, 3 object properties, and 51 data properties based on the dataset and expert's input were constructed in this work, and these new concepts and relationships proved to be valuable to the existing ontology.)

5.2.2 Ontology Modeling

The applied methodology for the ontology modeling in this work is comprised of the following four steps: (1) exploring and selecting an existing ontology, (2) determining new concepts from the underlying dataset to extend the ontology, (3) refining the results by consulting a sensory science expert, and (4) enriching the selected ontology. The ontology was selected based on the quality characteristics metric in (Hooi et al., 2015). This metric ensures that attention is paid to the quality description of the ontology in terms of classes, relationships, properties, and individuals. Considering the criterion metric, only one existing ontology was selected as the most suitable. This ontology combined information on both smell and taste disorders into a single ontology. Although some other ontologies, like Human Disease Ontology [5] and Medical Dictionary for Regulatory Activities Terminology (MedDRA) [6], have also smell and taste disorders as classes under their respective ontologies, robust information about the smell and taste disorders are not present in those ontologies. In contrast, the CSTD provides detailed information about the different types of smell and taste disorders, such as the etiology of these disorders, physical examination, and history of illness. Therefore, the CSTD was selected as the ontology to guide our understanding of relationships among different concepts of these disorders. It was retrieved from the Bioportal Bioontology website [7].

5.2.3 KG Construction

The ontology contains a formal representation of concepts, entities, and their relationships, but has limited reasoning capabilities to infer new knowledge without the data about instances. Instead, a KG can support acquiring knowledge by integrating information and providing flexibility to add newly derived knowledge on the go. KGs can be auto-generated or implemented with human involvement using semi-automatic or manual methods. They can be designed with an ontology or may be evolving with time and may have different shapes and sizes. A company, group of companies, or open-source community may be involved in the creation of KGs. Regardless of these distinctions, they aid in the organization of unstructured data so that information may be easily extracted, where the process is aided by explicit links between various elements.

[5]https://bioportal.bioontology.org/ontologies/DOID/
[6]https://bioportal.bioontology.org/ontologies/MEDDRA
[7]https://bioportal.bioontology.org/ontologies

To construct the KG, data were acquired from real patient data and integrated into the (revised by us) CSTD ontology. To bind a particular data source to a particular domain ontology, a mapping was established that connected individual data columns to entities from the ontology. For instance, a patient with number: 118190, gender: male, height: 1.79 m, weight: 89 kg, complaint duration: 2–5 years, etiology type: idiopathic, and BMI: 27.78 has taken the medicine Antiarhytmica. The same patient also has a smell disorder. A part of constructed KG is shown in Figure 5.1. The entities, such as patient number, height, weight, complaint duration, etiology type, and BMI value, are represented with nodes, while relationships between entities (e.g., the relationship between a patient ID and gender) are presented through edges. Each edge has a unique description of the relationships. For instance, the edge description of medication refers to which medicine has been taken by the patient. More detailed information about the construction of KG is described in Section 5.3.3.

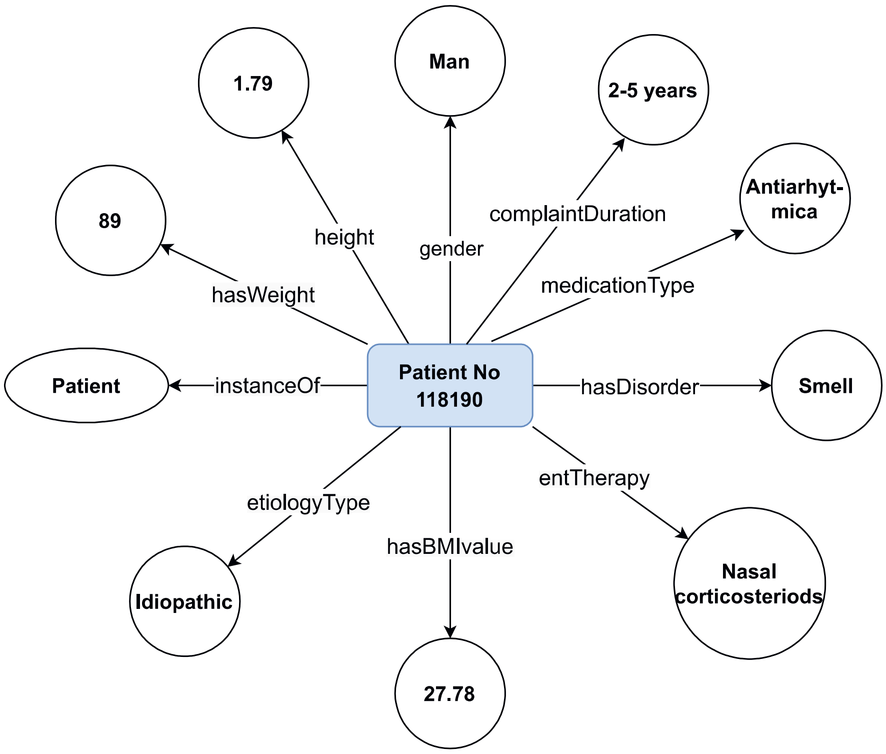


Figure 5.1: A sample of smell and taste disorders KG.

5.2.4 Question-Answering User Interface

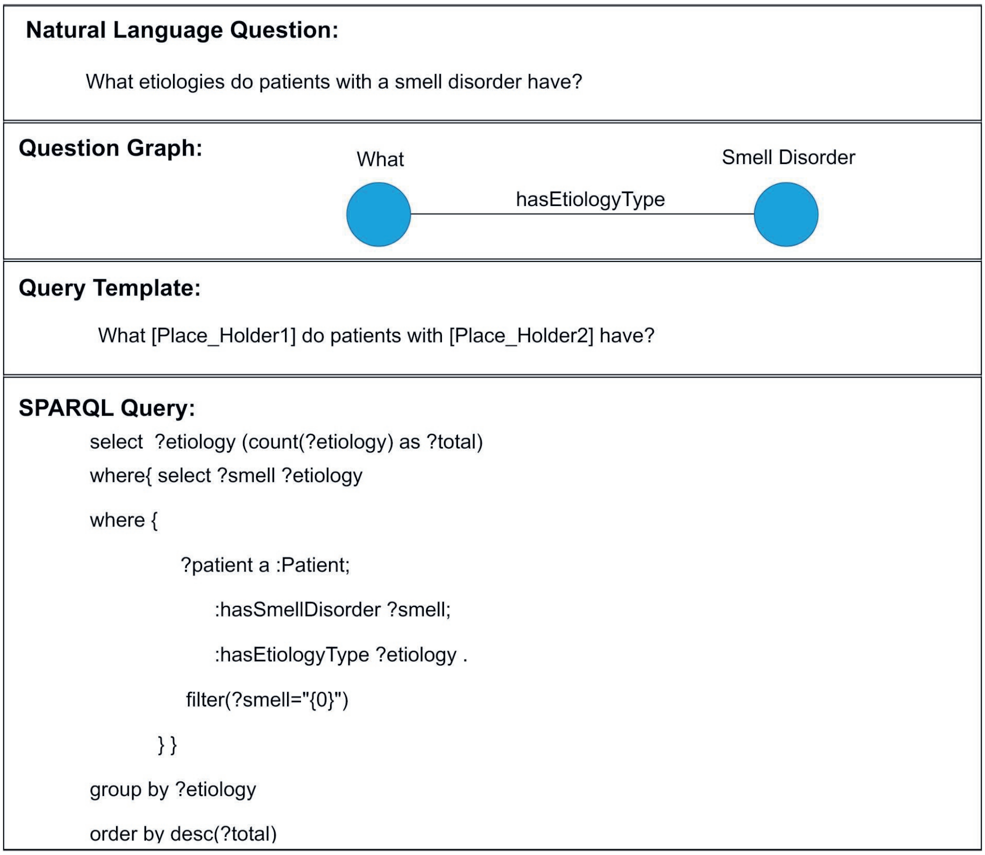
Writing a complex query is not advisable and practically manageable for users without a technical background. To support end-users and reduce the burden of query formulation, we developed an intuitive UI that empowers users by allowing them to access, discover, explore, and visualize the data and their relationships in a graph structure. The developed interactive UI targets bridging the gap between end-users and the knowledge (in RDF structure) stored in the KG. The targeted UI consists of two main components: (1) question selection and (2) KG visualization. In the question selection component, end-users select a pre-defined question from a drop-down menu and then change it based on their intentions.

The KG visualization component allows the user to see the structure of the schema (based on the chosen question in the QA component), including all primitive relations between entities. For example, a clinician wants to retrieve information about a specific patient's medications. In the following sections, we describe the detail of each component.

5.2.4.1 Question Selection

The idea behind the question selection component is to obtain a good compromise between intuitiveness and expressivity (Unger et al., 2012). This component exploits the expressiveness of the RDF KG and SPARQL while hiding their complexity.

To this end, we created a set of query templates that capture the semantic structure of the user questions, including specific slots for entities and classes. Let's consider "What [Place_Holder1] do patients with [Place_Holder2] have?" as a query template. The query templates are instantiated with actual questions such as "What etiologies do patients with a smell disorder have?" by replacing the slots "Place_Holder1" and "Place_Holder2" with "etiology" and "smell disorder", respectively. We predefined several possible questions based on the query templates that are shown in a drop-down menu to the users. Table 5.4 tabulates the list of predefined questions. Since there is a query template behind each predefined question, there is a possibility to create new questions by substituting the values of the slots. We took advantage of graphs as an intermediate form between query templates and questions to ease the process of substituting the values for the user. An intuitive graph is shown to users for each question so that the users can replace the nodes of the graph with their needs (such as the shown Question Graph in Figure 5.2). Moreover, the UI offers a list of all possible values for each node according to the stored facts in the KG. These values are basically verbalized forms (explained in Section 5.2.4.2) of entities or classes. Next, the slots of the template are filled with selected values, and finally, a SPARQL query is executed to retrieve the answer. Figure 5.2 presents how the gap between users' questions (i.e., unstructured) and SPARQL queries (i.e., structured) is narrowed down.



Natural Language Question:

What etiologies do patients with a smell disorder have?

Question Graph:

Query Template:

What [Place_Holder1] do patients with [Place_Holder2] have?

SPARQL Query:

```
select  ?etiology (count(?etiology) as ?total)
where{ select ?smell ?etiology
where {
          ?patient a :Patient;
              :hasSmellDisorder ?smell;
              :hasEtiologyType ?etiology .
           filter(?smell="{0}")
        } }
group by ?etiology
order by desc(?total)
```

Figure 5.2: Bridging gap between questions and SPARQL queries.

5.2.4.2 Knowledge Graph Visualization

KG visualization provides a way to reflect the true nature of the domain and the semantic relationships between concepts and entities. Moreover, it helps users to get insight from different classes, properties, and individuals. Since the target UI users are not experts in the RDF data models, we applied some simple but effective techniques to prune and simplify the underlying KG in the visualization process. The applied techniques are summarized as follows:

- Filtering unnecessary properties out: Those predicates that are not decisive in the context of QA over KGs (e.g., the properties applied to link the same entities between the underlying KG and other KGs or the properties used to provide data provenance) are refined (Aghaei et al., 2022a).
- Filtering entities out: Since a KG can include a number of triples showing a

relation between various entities (e.g., a specific medication is taken by several patients) and the purpose is to provide an understanding of the underlying KG and the relations, we show only a limited number of facts with the same relations.

- Verbalizing entities and properties: To make RDF triples more readable for end-users, the properties and entities with Uniform Resource Identifier (URI)s are verbalized. The process of verbalization includes using the name (or label) of entities and proprieties to present the labels of the edges and nodes. In addition, in the case there is no label or name for a given property or entity, we use the variable part of its URI (e.g., the URI "http://example.com/base/hasEtiologyType" as the predicate of a fact is verbalized to the human-readable term "ex:hasEtiologyType") (Aghaei et al., 2022c).

5.3 Experiments and Results

This section contains details about the experimental study and the results. In Section 5.3.1, the implementation setup is described. Section 5.3.2 and Section 5.3.3 explain ontology modeling and KG construction, respectively. The development of the UI is discussed in Section 5.3.4 and the result of the UI case study is presented in Section 5.3.5. Further, the evaluation of smell and taste disorder ontology is presented in Section 5.3.6.

5.3.1 Experimental Setup

We summarized the libraries, software, and frameworks used in this implementation in Table 5.1. GraphDB was selected as a database management system (DBMS) because of its capabilities, such as more intuitive data representation, storage, and management. To deploy the service layer, a system with 32 GB (gigabyte) random-access memory (RAM), a 1.7 gigahertz (GHz) AMD Ryzen 7 PRO 4750U processor, and 1 terabyte (TB) storage was used. Moreover, Linux with variant distributions, such as Ubuntu and Debian, was used for all other deployment setups.

Table 5.1: Libraries, frameworks, programming languages, and DBMS that were used for this research.

Libraries	Python Requests 2.25.1	React 18.2.0	Bootstrap 5.2.0	SQLite 2.6	SPARQL-Wrapper 1.8.5
Framework	Flask 1.1.2	Flask-RESTful 0.3.8	Flask-SQLAlchemy 2.5.1	Protégé 5.5.0	Plotly dash 2.6.1
Programming Language, DBMS	Python 3.8	GraphDB free edition 9.4.1			

5.3.2 Ontology Modeling

After selecting the CSTD ontology, the classes, individuals, and relationships were thoroughly further examined using Protégé [8]. Protégé is an open-source ontology editor and framework for building semantic data structures and it provides an effective user interface for examining and editing ontologies with their respective annotations. Moreover, OntoGraph (a Protégé plugin) was applied to visualize and examine the class and relationship hierarchies. CSTD originally consisted of four classes including chemosensory phenotyping, smell disorders, and taste disorders. The chemosensory phenotyping class has sub-classes including allergy evaluation, demographic characteristics, family history, chemosensory symptoms or signs, history of illness, physical examination, review of symptoms, and social history. While examining CSTD, we discovered it did not follow the best ontology engineering practices. For example, the CSTD ontology naming conventions were inconsistent. Therefore, we opted to create a new ontology. The new ontology is created taking the information presented in CSTD as a base and improving it following the best practices of ontology engineering. A manual investigation was done by comparing the concepts in the CSTD with the data available at the Smell and Taste Centre database (see details in Section 5.2.1). The comparison showed that the CSTD ontology had to be extended with new concepts. These new concepts were divided into seven classes, namely, social demographics, medication, therapy, assessment, improvement, medical condition, and diagnosis. Each class and its respective sub-classes are shown in Table 5.2. Finally, we validated the categorization of the added concepts by consulting with a sensory science expert before embedding them into the new ontology model. In addition, an overview of the extended ontology with respect to smell and taste disorders is shown in Figure 5.3. The added classes including the code used in the development process are publicly available on GitHub (Tauqeer, 2022).

[8]https://protege.stanford.edu/

Table 5.2: Newly added concepts in the ontology.

Social Demographic Characteristics	**Medication**	**Therapy**	**Assessment**	**Improvement**	**Medical Condition**	**Diagnosis**
Weight	Stomach medication	ENT Therapy	Sniffin Sticks Threshold	Timing for follow-up	Parosmia	Retronasal test diagnosis
Length	Pain medication	Smell Therapy	Sniffin Sticks Discrimination	Improvement Variable	Anamnestic smell or taste disorder	Taste Strips diagnosis
BMI	Osteoporosis medication		Sniffin Sticks Identification			
	Antidiabetics		Sniffin Sticks TDI			
	Antiarrhythmics		Retronasal test			
	Antihypertensives		Taste Strips			
	Antibiotics		MRI test			
	Anticoagulants					
	Antiallergics					

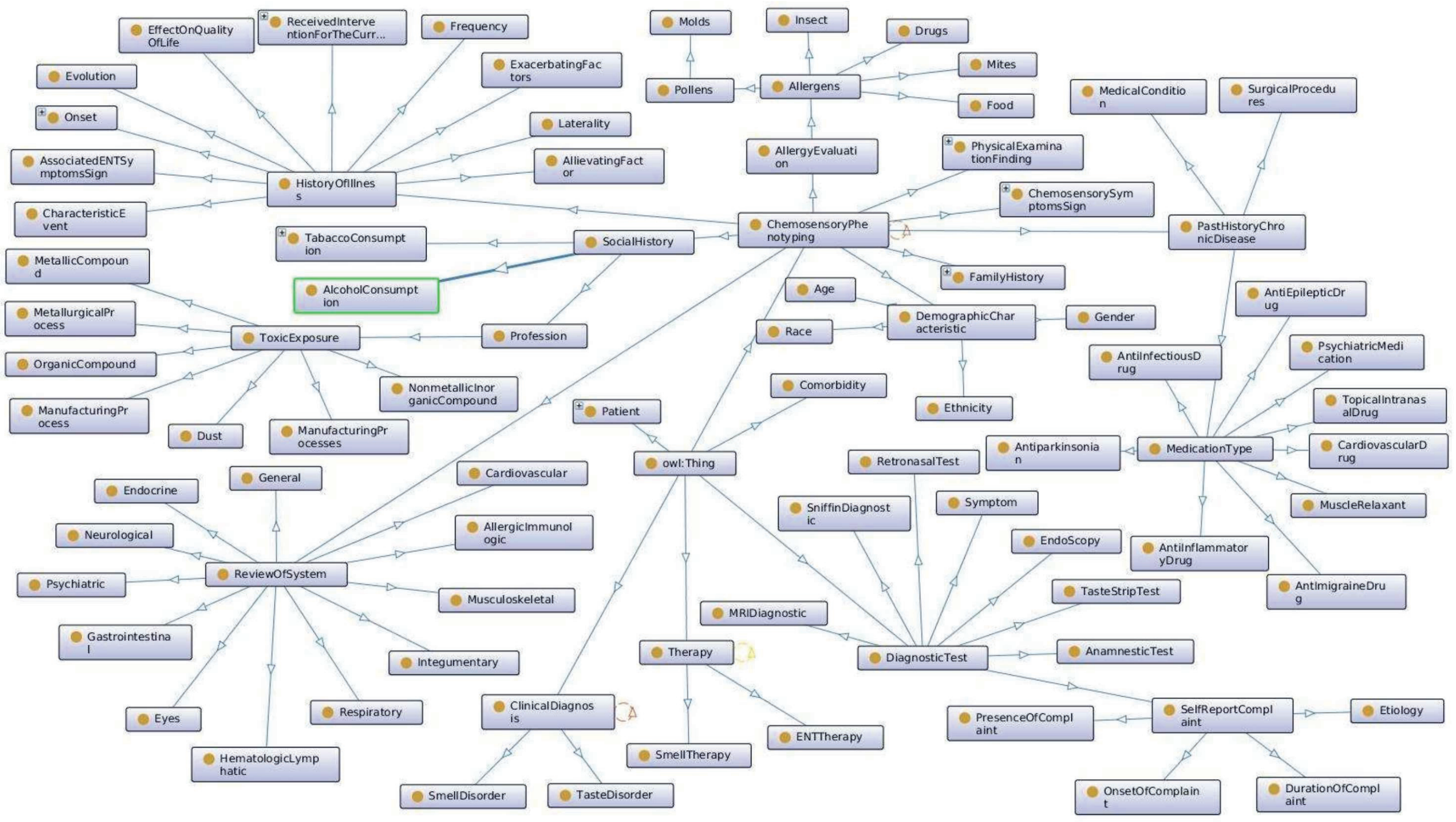


Figure 5.3: An overview of the extended ontology for the smell and taste disorder.

5.3.3 KG Construction

For the construction of KG, a GraphDB instance, and to map tabular data, the GraphDB OntoRefine tool was used. The latter is an open-source data transformation tool that allows a quick mapping of structured data to a locally stored RDF schema in GraphDB. A manual approach was used for this mapping. In (Kalaycı et al., 2020) a mapping is defined as “a set of assertions specifying how the data from the sources populate the classes and properties of the ontology”. The components of each mapping assertion are an identifier (e.g., a patient number is an identifier and also an instance of class *ex: Patient* - see Table 5.3) (Kalaycı et al., 2020). A snapshot containing a part of the ontology schema mapping through the OntoRefine tool is presented in Table 5.3.

Table 5.3: An example of ontology schema mapping using OntoRefine.

Subject	**Predicate**	**Object**
Patient_Nr <IRI>	a <IRI>	Patient <IRI>
	schema: gender <IRI>	Gender “Literal”
	hasSmellDisorder <IRI>	Retronasal_ahn “Literal”
	hasTasteDisorder <IRI>	Tastedisorder “Literal”
	hasComplaintDuration <IRI>	Duration_of_complaint “Literal”
	hasEtiologyType <IRI>	Etiology “Literal”
	hasComorbidity <IRI>	Comorbidities “Literal”
	hasEntTherapy <IRI>	ENT_therapy “Literal”
	hasSmellTherapy <IRI>	Smell_therapy “Literal”
	hasAntiAllergic <IRI>	Antiallergica “Literal”
	hasAsthmaMedication <IRI>	Astmamedicatie “Literal”
hasSmellDisorder <IRI>	a <IRI>	rdfs: Property <IRI>
		owl: DatatypeProperty <IRI>
	rdfs: domain <IRI>	Patient <IRI>
	rdfs: range <IRI>	xsd:string “Datatype”
hasTasteDisorder <IRI>	a <IRI>	rdfs: Property <IRI>
		owl: DatatypeProperty <IRI>
	rdfs: domain <IRI>	Patient <IRI>
	rdfs: range <IRI>	xsd:string “Datatype”

IRI: Internationalized Resource Identifier
rdfs: Resource Description Framework Schema
owl: Web Ontology Language

To illustrate ontology schema mapping with tabular data, an example of a patient with smell and taste disorders has been taken. A *Patient_Nr* is an individual of class *ex:*

Patient while *ex:patientNo* is a data property. Two data properties *ex:hasSmellDisorder* and *ex:hasTasteDisorder* of class *ex:SmellDisorder*, *ex:TasteDisorder* respectively were defined. Pre-built properties such as *rdfs:Property* and *owl:ObjectProperty* were used to associate properties with their respective classes. For each object property, there is a need to define *rdfs:domain* and *rdfs:range*. The domain property refers to the subject's value, while the range property, specifies a value regarding the object's property. For instance, the *ex:hasSmellDisorder* is a type of data property, which has domain *ex:Patient* class, and range *exd:string*. Similarly, we applied the same procedure to map other classes and properties. Once the ontology schema has been mapped, the data has been extracted and imported to the Protégé to create KG with new concepts. All new classes (see Table 5.2) are created in Protégé with descriptions, language tags, domains, and ranges. Figure 5.4 displays the semantic representation of a selection of concepts within the newly developed KG.

This representation provides a visual illustration of the interrelationships among the concepts within the graph, thereby aiding in the understanding and interpretation of the data contained therein. It should be noted that the semantic representation presented in Figure 5.4 does not encompass all the concepts within the KG. Rather, it offers a selection of the key concepts and their interrelationships for the purpose of providing a clear and concise illustration of the underlying structure of the graph.

The classes that have an is-a relationship with *owl:Thing* are shown on gray backgrounds while sub-classes are presented on sky-blue backgrounds. In Protege, 21 new classes such as *ex:DurationOfComplaint*, *ex:Therapy*, *ex:Comorbidity*, *ex:Etiology*, *ex:ClinicalDiagnosis*, *ex:DiagnosticTest*, and *ex:Patient* were created. To use these classes to get the answers over KG, 3 new object properties (*ex:typeDisorder*, *ex:typeTherapy*, and *ex:chemosensoryPhenotyping*) and 51 data properties (e.g., *ex:hasEntTherapy*, *ex:hasLength*, *ex:hasWeight*, *ex:hasDurationOfComplaint*) were constructed. Relationships between classes and properties were defined via the domain and range properties. The newly constructed KG with mapping information and source code is available on GitHub (Tauqeer, 2022). The constructed KG consists of 11,093 statements (10,763 explicit and 330 inferred). The KG is stored in a GraphDB instance and can be accessed via a SPARQL endpoint at ("SPARQL, Endpoint URL", 2023). Furthermore, to access the tool the URL https://disorder-question-answer-interface.sti2.at/ can be used, and it is also publicly available. All the source code and material are also available at GitHub (Tauqeer, 2022)

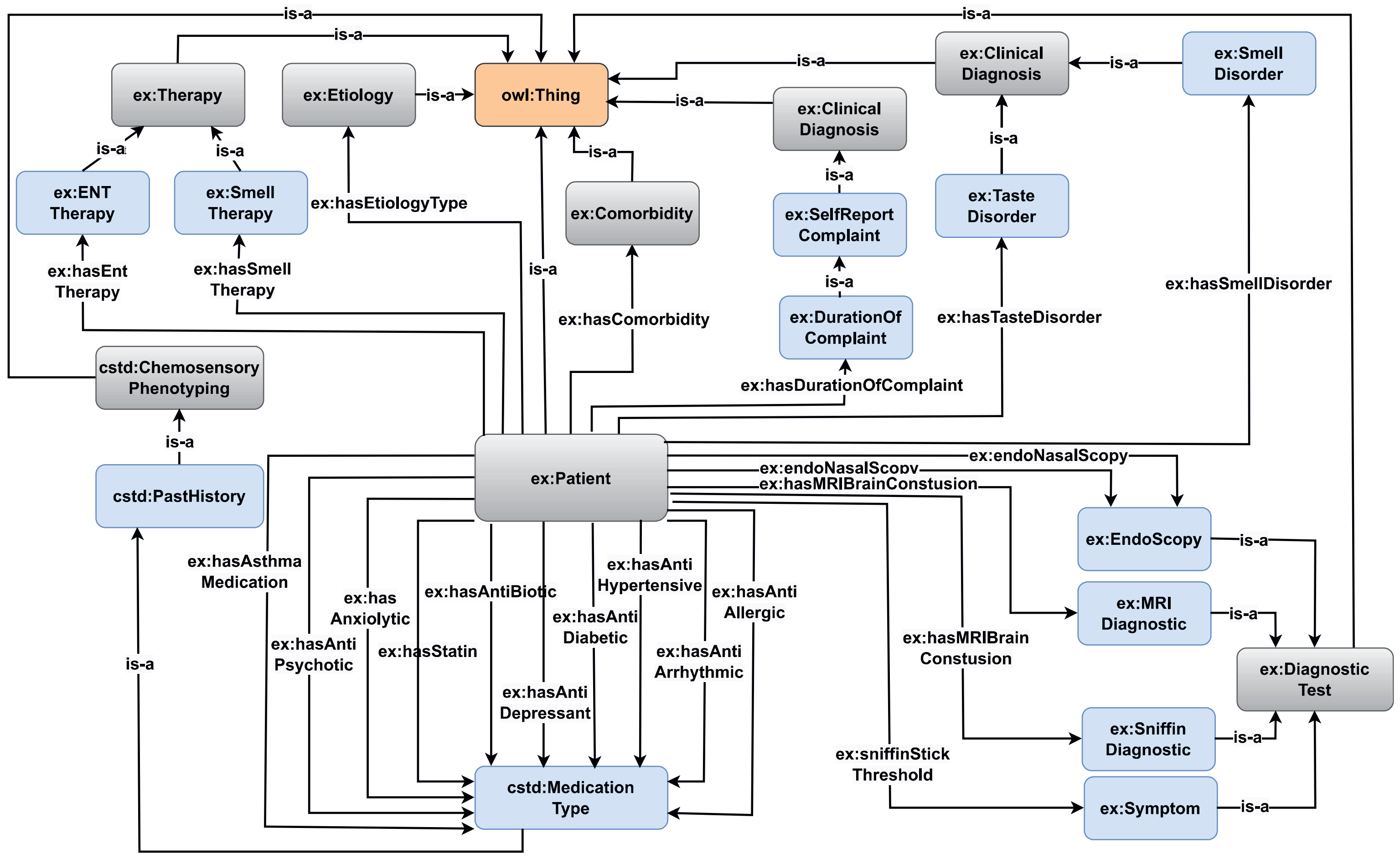


Figure 5.4: A semantic representation of KG (selected extended concepts).

5.3.4 UI Development

A browser-based UI is developed to ease end-users access to information and can be accessible via https://disorder-question-answer-interface.sti2.at/. The libraries and frameworks that were used to develop the UI are described in Table 5.1. The UI supports features like question graph creation, updating the previously created question graph, the visualization of KG, and displaying answers to the selected questions. Since a KG is stored on GraphDB (i.e., graph database) a connection is required to access and store information from KG. A SPARQL endpoint is used for that purpose. The main interface of question answering is presented in Figure 5.5.

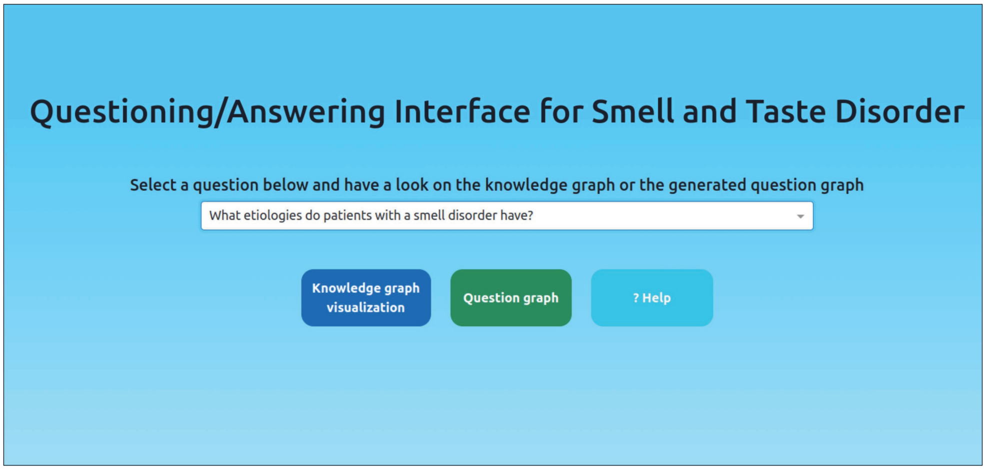
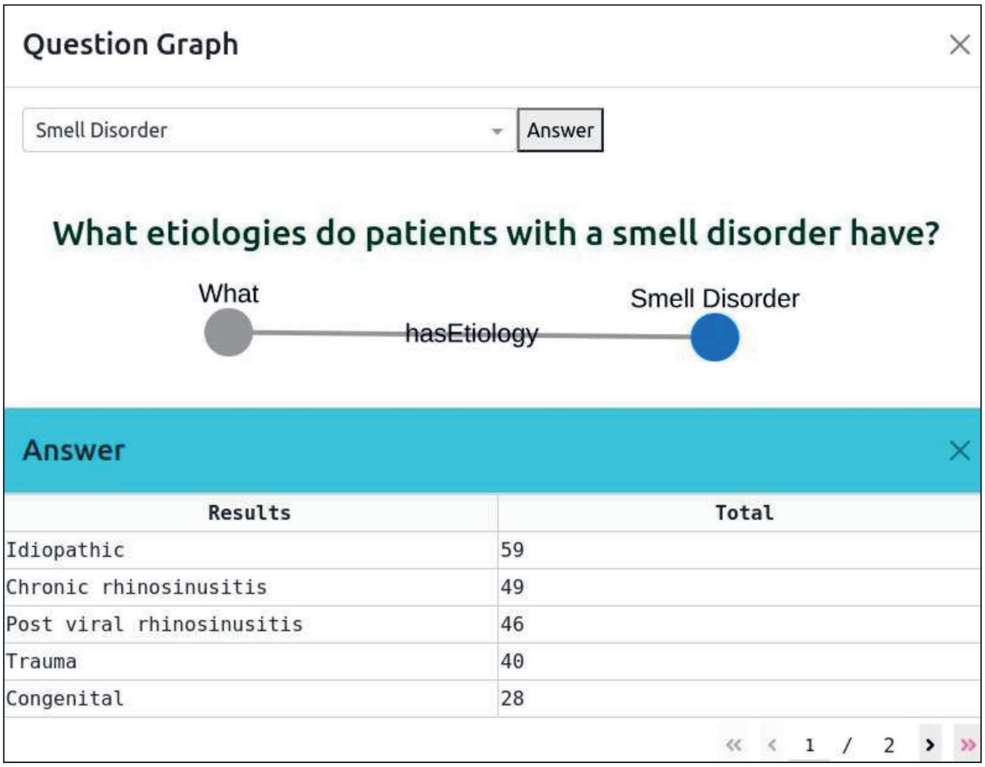

Questioning/Answering Interface for Smell and Taste Disorder
Select a question below and have a look on the knowledge graph or the generated question graph
What etiologies do patients with a smell disorder have?
Knowledge graph visualization
Question graph
? Help

Figure 5.5: Question answering UI.

The developed UI consists of a dropdown menu and three buttons labeled "Knowledge graph visualization", "Question graph", and "Help". The dropdown menu contains formulated questions (see Table 5.4) designed by domain experts. End-users can select the question from the dropdown menu whenever they visit the UI. The buttons labeled "Knowledge graph" and "Question graph" can not work unless a question is selected from the dropdown menu. Each time an end-user clicks on the "Question graph" button a new window will appear containing a question graph template (graph representation of question), a question text, a button labeled with the answer, and a dropdown menu as presented in Figure 5.6.

End-users can change disorder types (regarding smell and taste disorders) through the dropdown menu. We have implemented a query template for each predefined question. For instance, we took question 1 "What etiologies do patients with a smell disorder have?" as an example. The query template for question 1 is shown in Listing 5.1.

Listing 5.1: A query template for question 1.



What etiologies do patients with a smell disorder have?

Results	Total
Idiopathic	59
Chronic rhinosinusitis	49
Post viral rhinosinusitis	46
Trauma	40
Congenital	28

Figure 5.6: Overview of questions graph layout.

```
PREFIX : <http://example.com/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select ?hasEtiologyType (count(?hasEtiologyType) as ?total)
where{
    select *
    where {
        ?patient a :Patient;
              :hasSmellDisorder ?smell;
              :hasEtiologyType ?hasEtiologyType .
         filter(?smell="{param}")
    }
}
group by ?hasEtiologyType
order by desc(?total)
```

The SPARQL select query extracts the patients with a smell disorder having etiology types (e.g., “Trauma”). Additionally, data is filtered based on smell disorder “anosmia” for instance. Finally, it returns etiology types with their total in ascending order.

The button labeled with the answer is used to display the results of the current selected question as presented in Figure 5.6. More information about the question’s results can be found in Section 5.3.5. End-users can select types of smell or taste disorders (e.g., anosmia or hyposmia) from the drop-down menu once they have selected the node to change. We used this changed value as a parameter shown in Listing 5.1.

The end-users can get an insight into constructed KG by clicking on the button labeled “Knowledge graph visualization”. A new window containing a subset of constructed KG will appear as shown in Figure 5.7.

It shows different properties and individuals. Since end-users are not experts in the RDF data models, the underlying KG in the visualization is kept simple by (1) filtering out unnecessary properties, (2) filtering out entities, and (3) verbalizing entities and properties. Additionally, various colors and shapes have been introduced to distinguish between nodes, edges, and relationships. For instance, nodes (i.e., subject) are presented in circles in red color. Examples of these nodes are ex:patientNo 4550, ex:patientNo 5820, and ex:patientNo 280. The attribute values (i.e., objects) are presented in green color. “Surgery”, “Man”, “anosmia”, “Idiopathic”, and “86” are typical examples of objects. Lastly, the relationships (i.e., predicates) between subjects and objects are shown with edges (in light green color), while the labels are presented in blue color. Some examples of predicates are ex:hasBMI, ex:hasWeight, ex:hasLength, ex:hasAntiBiotic, and ex:hasAsthmaMedication. A concrete example of the subset of smell and disorder KG of three patients with patient numbers ex:patientNo 280, ex:patientNo 4550, and ex:patientNo 5820 is presented in Figure 5.7. Edges between nodes represent the relationship between them. For instance, the patient number ex:patientNo 280 has a complaint duration of *“10 years”* while patient number ex:patientNo 5820 has a complaint duration of *“3-24 months”*. Similarly, other relationships such as gender, medication type, length, and weight can be seen in Figure 5.7.

The button labeled “Help” is used to help the end-users with how they can interact with UI to make questions over KG and what types of features it has. Once end-users click on the “Help” button, they can see the help about the following UI features: (1) selection of a fixed-sized set of questions (see Figure 5.5), (2) visualizing a subset of a KG corresponding to the question (see Figure 5.7), (3) generating a question graph for the question (see Figure 5.9), and (4) editing the generated question graph.

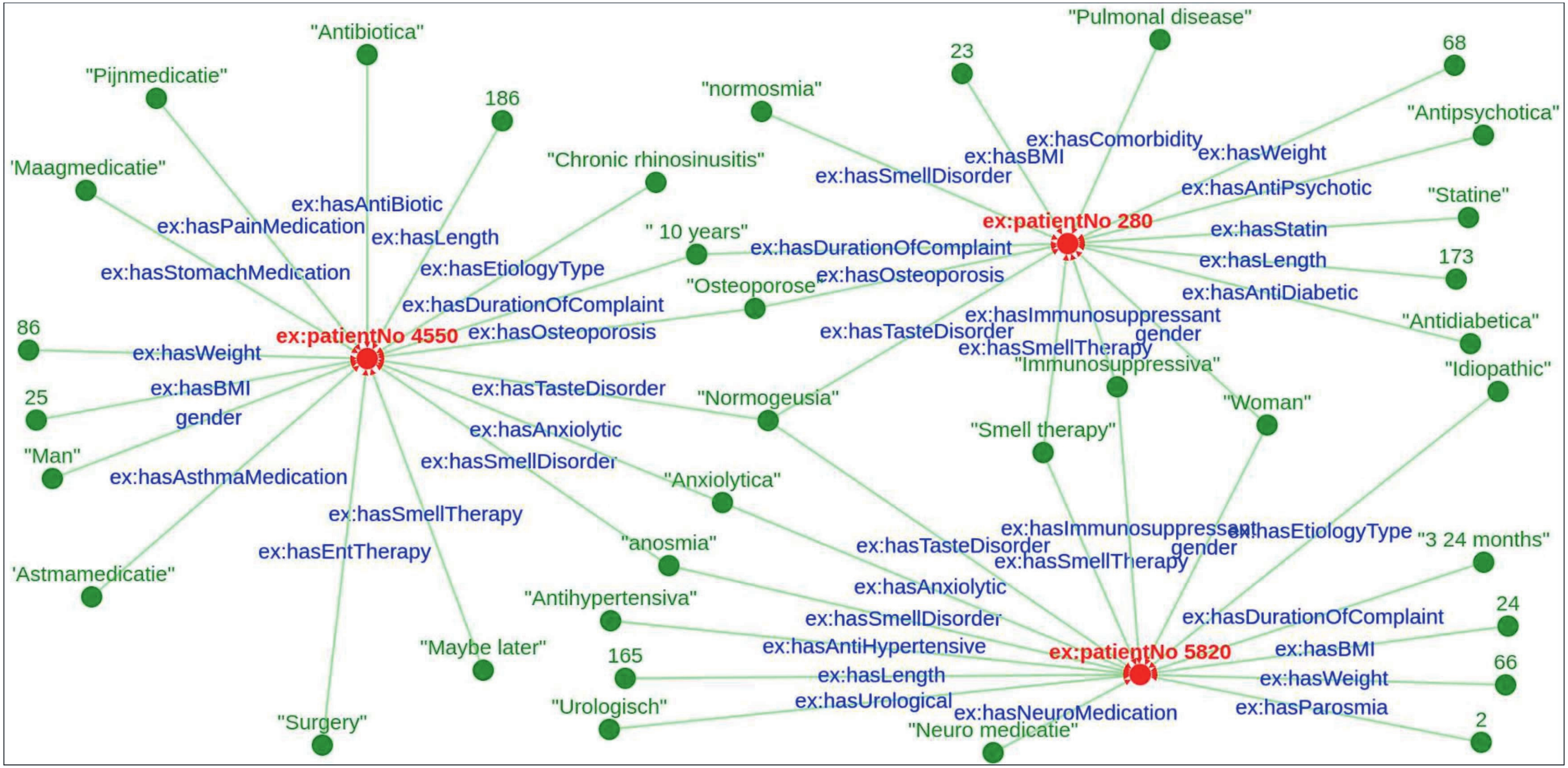


Figure 5.7: Subset of smell and taste disorder KG.

5.3.5 UI Case Study Result

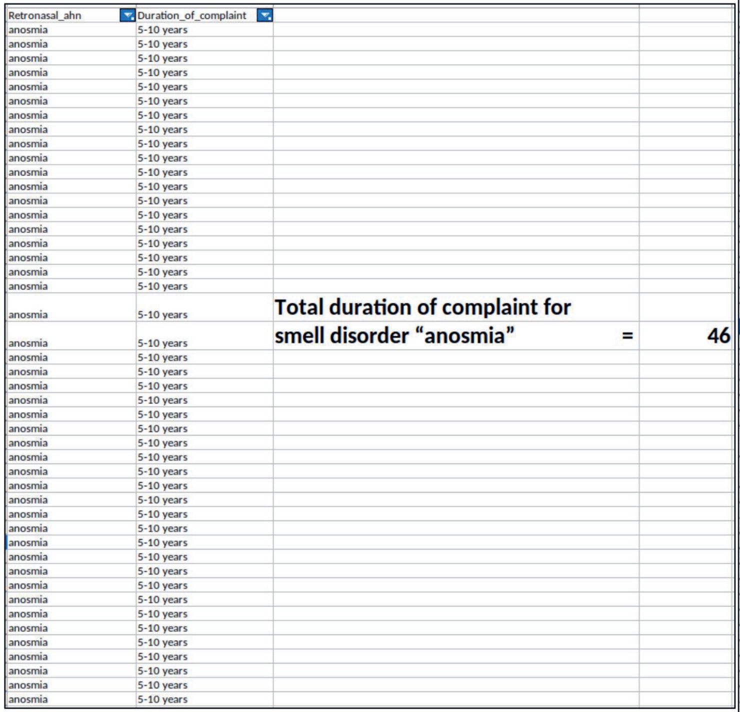
To demonstrate the functionality and usability of the tool, we conducted a case study to make use of the smell and taste disorder KG. For this aim, we asked a sensory science expert to formulate 8 questions related to smell and taste disorders. All questions are based on etiology, comorbidity, complaint duration, and medication type for smell and taste disorders. Later we asked the expert to find the related answers using our tool and check the validity of the answers. These results/answers were then validated manually against the original resource (i.e., dataset described in Section 5.2.1). Table 5.4 shows the formulated questions while the manual validation of the first and seventh questions results are presented in Figure 5.8.

There were 5 etiology types used in the dataset, and the result of question 1 also shows the same etiology types. The most dominant etiology type is *Idiopathic* with a total of 59 which can also be seen in the evaluation of our results (see Figure 5.8). On the other hand, 6 complaint durations were defined in the original dataset and our UI also showed the same complaint duration. The 7th question result shows the complaint duration (i.e., "5-10 years") with a total of 46 which can also be seen in Figure 5.8. The same procedure is followed for other results verification from the original resource.

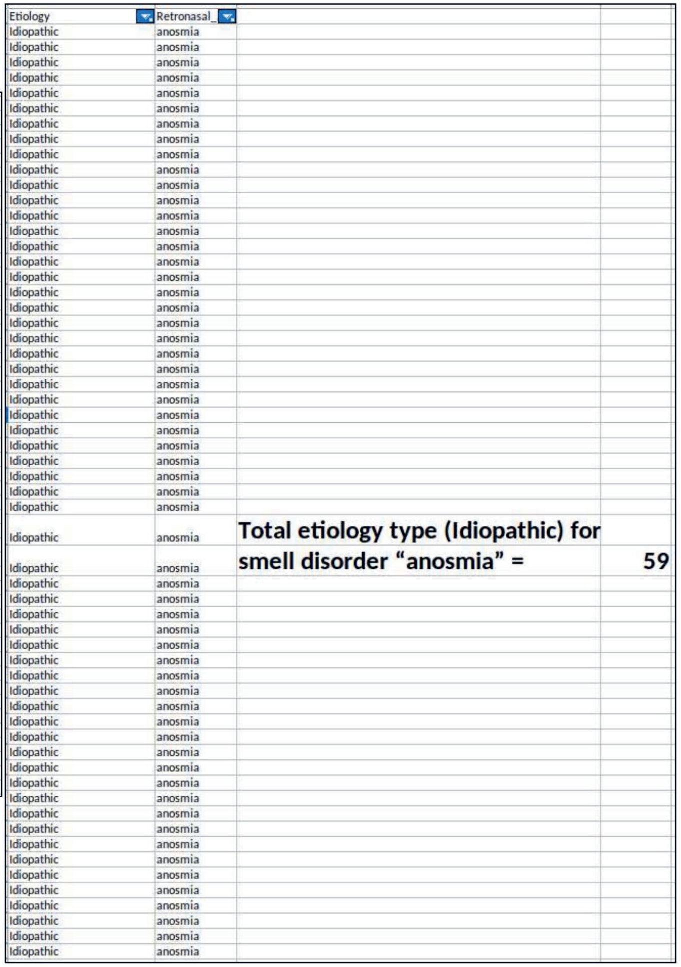
We took questions 1 and 7 as examples to show their results from UI over KG (see Figure 5.9). Both figures contain a question graph, a question text, and the result of the question. For instance, in KG, the most dominant etiology for patients with "anosmia" is *"Idiopathic"* (59 in total) while *"Iatrogenic"* is the least dominant. On the other hand, for patients with a smell disorder, the least reported complaint duration was "$<$ 3 months", while the most reported complaint duration was "2-5 years" as shown in Figure 5.9. Similarly, the rest of the questions have been answered correctly using the developed UI. The screenshots of each question with their result can be found at GitHub (Tauqeer, 2022).

Further, we asked the expert to try the KG visualization component and share the explored knowledge that can not be easily explored by just looking at the raw database. The expert was able to explore the following queries in this case study with the help of the KG visualization component:

- Find relationships between different etiologies.
- Comprehend disorders hierarchy.
- Explore hidden relationships in the data.



Retronasal_ahn	Duration_of_complaint		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years	**Total duration of complaint for**	
anosmia	5-10 years	**smell disorder “anosmia” =**	**46**
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		
anosmia	5-10 years		



Etiology	Retronasal_		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia	**Total etiology type (Idiopathic) for**	
Idiopathic	anosmia	**smell disorder “anosmia” =**	**59**
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		
Idiopathic	anosmia		

Figure 5.8: A manual result validation of questions 1 and 7.

Table 5.4: Formulated experimental questions.

No	Questions
1	*What etiologies do patients with a smell disorder have?*
2	*What is the most used medication for patients with a smell disorder?*
3	*What etiologies do patients with a taste disorder have?*
4	*What is the most used medication for patients with a taste disorder?*
5	*What are the most common comorbidities for patients with a smell disorder?*
6	*What are the most common comorbidities for patients with a taste disorder?*
7	*What is the minimum complaint duration for a smell disorder?*
8	*What is the most common duration for a taste disorder?*

Figure 5.7 displays information about disorders and their etiology types regarding patients. In addition to the disorders, the graph also shows different types of medications used by patients with smell and taste disorders. The tool is able to visualize each specific disorder in broader disorder categories which makes it easier for the end-user to understand the smell and taste disorder hierarchy. Another hidden relationship that can be seen here is the connection of the nodes to the patient via ex:patientNo 280. It can be inferred that both patients number ex:patientNo 4550 and ex:patientNo 5820 had a smell disorder (or "anosmia") as a common disorder.

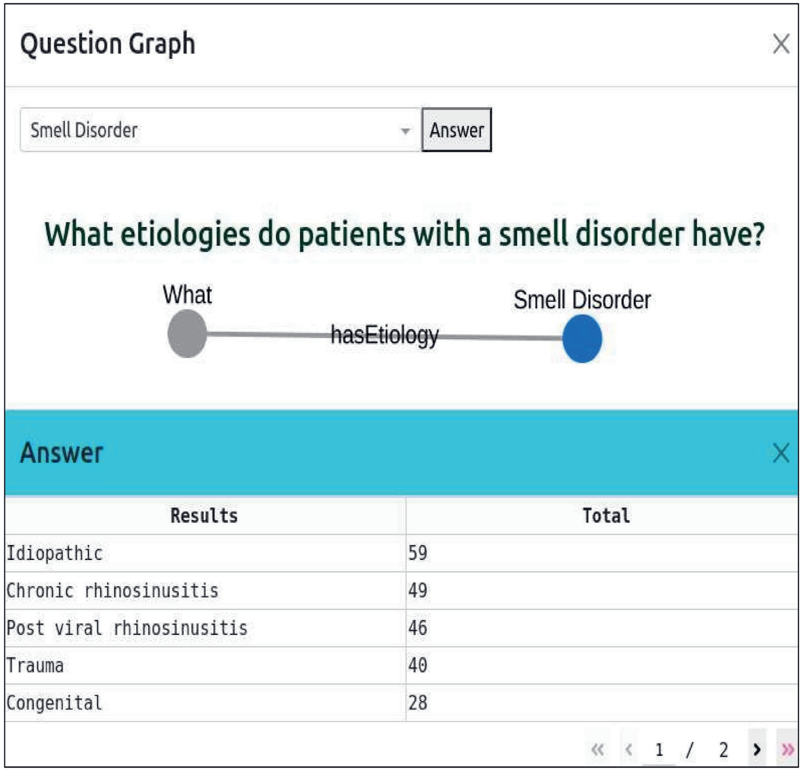
It is fair to conclude that the tool effectively achieves all intended outcomes, as our UI was able to answer all the expert questions and the KG visualization was able to find the hidden relationships between the data points, easily classify them based on their relationship, and explore already existing relationships. However, the efficiency of the tool should be further investigated when it comes to KGs with a larger number of nodes.

5.3.6 KG and Ontology Evaluation

After smell and taste disorder KG creation and enrichment, a crucial step is to perform a quality assessment of KG for its evaluation. This evaluation is typically conducted along four primary dimensions, namely *"Accuracy"*, *"Coverage"*, *"Coherency"*, and *"Succinctness"* (Hogan et al., 2021). The degree to which entities and relations, conveyed by nodes and edges in the graph, accurately replicate real-life phenomena is referred to as *"Accuracy"*. Given that the smell and taste disorder KG is developed collaboratively with domain and ontology experts, it has successfully passed the *"Syntactic Accuracy"* check which verifies that the data adhere to the grammatical norms established for the domain and/or data model. Moreover, it has also cleared *"Semantic Accuracy"* check which measures the degree of semantic correctness by assessing how precisely data values reflect actual phenomena. It can be impacted by erroneous extraction results, suspect sources, vandalism, and the like. The KG has been adequately represented with real-life phenomena to pass this check. Further, the Pellet and HermiT reasoners were employed to evaluate both the *"Coherent"* and *"Consistency"* of the ontology (see Figure 5.10). *"Completeness"* and *"Representativeness"* are two parts of *"Coverage"* quality assessment dimension. The former pertains to the degree in which all required information is present in a particular dataset. While the latter is focused on assessing high-level biases in what is included or excluded from the KG as highlighted by Baeza-Yates (Baeza-Yates, 2018).

Drawing from the original dataset, the present study conducts an assessment of two key dimensions of *"Completeness"* in the extended KG. The first dimension, schema completeness, pertains to the extent to which the classes and properties of a given schema are represented in the data graph. The second dimension, property completeness, concerns the degree to which a specific property is complete with respect to the proportion of missing values. Another essential aspect of quality assessment is *"Coherency"* which encompasses the degree of adherence of the KG to the formal semantics and constraints defined at the schema level, as previously discussed by Hogan et al. (Hogan et al., 2021). *"Consistency"* and *"Validity"* constitute the two sub-dimensions of *"Coherency"*. The former sub-dimension verifies that the KG does not contain any contradictions, while the latter sub-dimension ensures that the KG is free of constraint violations. Since we employ a predefined query template for the questions/answer interface, no constraints are applicable. Finally, the fourth and final dimension of quality assessment, *"Succinctness"* ascertains that the KG contains only relevant content, thereby preventing information overload.

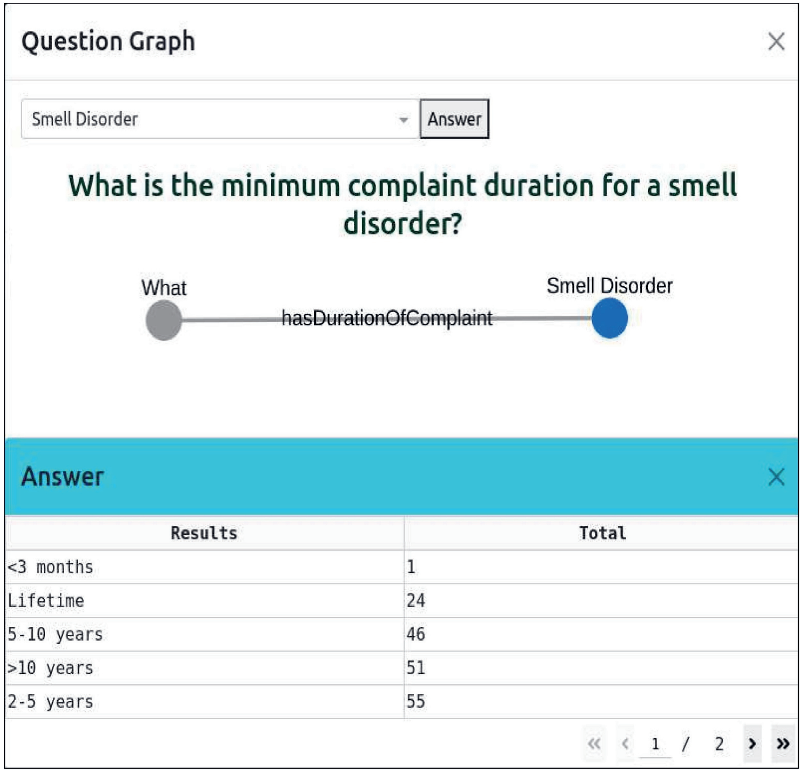
The *"Coverage"* quality assessment dimension is further subdivided into three categories, namely *"Conciseness"*, *"Representational Conciseness"*, and *"Understandability"*. Ensuring that the schema and data items are germane to the domain is considered to be concise. Redundancy cannot exist in the schema's classes, properties, among other elements. Similarly, redundant entities and relationships cannot be described in the data. The degree to which the content is presented concisely in the KG, either intentionally or extensionally, is referred to as representational conciseness (Zaveri et al., 2012). The third category, *"Understandability"*, pertains to the understandability of the data by humans, which should include legible labels and descriptions (preferably in multiple languages) that humans can comprehend without difficulty, as indicated by Kaffee et al. (Kaffee et al., 2017). In the case of the smell and taste disorder KG, the labels and descriptions employed are readily understandable. Furthermore, the ontology underwent evaluation with the OOPS! pitfall scanner (Poveda-Villalón et al., 2014).



What etiologies do patients with a smell disorder have?

Results	Total
Idiopathic	59
Chronic rhinosinusitis	49
Post viral rhinosinusitis	46
Trauma	40
Congenital	28

(a)

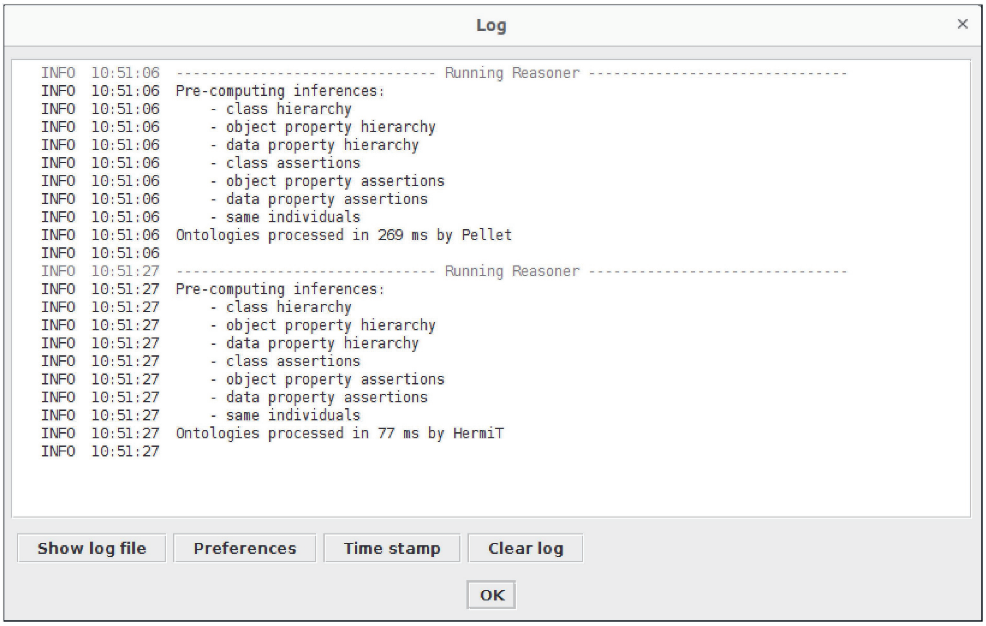


What is the minimum complaint duration for a smell disorder?

Results	Total
<3 months	1
Lifetime	24
5-10 years	46
>10 years	51
2-5 years	55

(b)

Figure 5.9: Results of the showcase questions 1 and 7. (a) Question 1. (b) Question 7.



```
INFO  10:51:06  ------------------------------ Running Reasoner ------------------------------
INFO  10:51:06  Pre-computing inferences:
INFO  10:51:06      - class hierarchy
INFO  10:51:06      - object property hierarchy
INFO  10:51:06      - data property hierarchy
INFO  10:51:06      - class assertions
INFO  10:51:06      - object property assertions
INFO  10:51:06      - data property assertions
INFO  10:51:06      - same individuals
INFO  10:51:06  Ontologies processed in 269 ms by Pellet
INFO  10:51:06
INFO  10:51:27  ------------------------------ Running Reasoner ------------------------------
INFO  10:51:27  Pre-computing inferences:
INFO  10:51:27      - class hierarchy
INFO  10:51:27      - object property hierarchy
INFO  10:51:27      - data property hierarchy
INFO  10:51:27      - class assertions
INFO  10:51:27      - object property assertions
INFO  10:51:27      - data property assertions
INFO  10:51:27      - same individuals
INFO  10:51:27  Ontologies processed in 77 ms by HermiT
INFO  10:51:27
```

Figure 5.10: Evaluation results of the ontology using Pellet and HermiT reasoners in Protege.

The OOPS! pitfall scanner is a widely used tool for detecting and evaluating ontology pitfalls (41 in total) classified in dimensions like structural, functional, and usability. In this study, the ontology was evaluated using this tool to identify any potential issues that could affect its quality. The results showed that there were no critical issues, which indicates that the ontology is of high quality. Eventually, the ontology can be used under license “Attribution 4.0 International (CC BY 4.0) [9]”.

5.4 Discussion and Conclusion

In this study, we have successfully enriched an existing ontology on smell and taste disorders and constructed a KG by integrating real anonymised patients' data. To the best of our knowledge, this is the first attempt to enrich an ontology and build a KG for smell and taste disorders. The quality of the KG was evaluated and a question-answering user interface (UI) was developed as a proof-of-concept to showcase a possible application of the KG. Through a case study, we have shown that our KG can facilitate the accessibility and usability of healthcare data in daily practice. Our findings show that visualizing data in the form of a graph makes it more tangible, intuitive,

[9]https://creativecommons.org/licenses/by/4.0/

and valuable to end-users. It provides a general overview of the domain by showing different concepts, classes, properties, and relationships between them. By using the developed UI, the end-user can find answers to their questions easily by selecting a question from the menu and modifying it by clicking on the generated question graph to change its components. The results of this study indicate that our techniques can significantly improve the accessibility and usability of healthcare data, which is crucial for patients with smell and taste disorders.

The semi-automatic enrichment of the openly available CSTD ontology provides a significant advantage in our work. We added 21 new classes, 3 object properties, and 51 data properties to the ontology, making it more representative of smell and taste disorders in general. Transforming data into a KG and making it accessible with a UI offers a better understanding of the association between patients' characteristics (e.g., age, gender, and BMI) and smell and taste disorders.

However, there are limitations to this study, such as the absence of data related to smell and taste disorders caused by Covid-19 as the data used for ontology enrichment were collected before the Covid-19 pandemic. Therefore, the information related to smell and taste disorders associated with Covid-19 as etiologies is missing from the current ontology. Also, the current dataset was collected at the Dutch Smell and Taste Center. However, there are also other patient groups that experience smell and taste disorders but do not visit this center, like cancer patients undergoing chemotherapy (Cohen et al., 2016). Therefore, the enriched ontology should be updated further with the latest developments in regard to smell and taste disorders by using newly collected data from different clinical centers. Moreover, the UI was developed with the aim of improving the accessibility and usability of the knowledge graph for healthcare professionals. However, this was only a proof-of-concept to show a possible application of the ontology and KG presented in this work. Before its use in daily practice, this UI should be evaluated on its effectiveness in achieving these goals among further health care professionals. This will be done in future research. Future work will focus on updating the CSTD ontology to include the latest developments related to smell and taste disorders, including data from other sources. Furthermore, usability testing of the UI will be conducted to evaluate its effectiveness and address any issues or concerns raised. We plan to interlink the smell and taste ontology with other ontologies, such as food ontologies, to enable a more comprehensive understanding of smell and taste loss. Additionally, we will incorporate GDPR data sharing with consent or contracts (Chhetri et al., 2022b; Tauqeer et al., 2022) with compliance for collecting smell and taste disorder data from patients. These developments will enhance the KG's applicability and utility in daily practice, providing better insights into the diagnosis and treatment of smell and taste disorders.

Funding

This study was partially supported by the Eureka, Eurostars Programme (grant agreement number 877857 with the Austrian Research Promotion Agency (FFG)) (work of Sareh Aghaei and Anna Fensel), as well as the Netidee scholarship 2022 by the Internet Foundation Austria (work of Sareh Aghaei).

CRediT authorship contribution statement

Amar Tauqeer: Conceptualization, Methodology, Software, Data Curation, Writing-Original Draft, Writing-Review Editing, Visualization. **Ismaheel Hammid:** Conceptualization, Methodology, Data Curation, Writing-Original Draft, Writing-Review Editing. **Sareh Aghaei:** Methodology, Data Curation, Writing-Review Editing, Visualization. **Parvaneh Parvin:** Conceptualization, Methodology, Writing-Original Draft, Writing-Review Editing, Resources. **Elbrich M. Postma:** Conceptualization, Methodology, Writing-Review Editing, Resources. **Anna Fensel:** Conceptualization, Methodology, Supervision, Project administration, Funding acquisition, Writing-Review Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability Statement

The dataset used for this study is available at the Smell and Taste Centre, Ede, the Netherlands, upon request by sending an email to reukensmaak@zgv.nl.

Acknowledgments

We acknowledge the contribution of Sanne Boesveldt, who shared her expert knowledge regarding the development of the new data model for ontology. Further, we thank Simon Kleinfeld, Marco Salchner, and Manuel Buchauer for providing the initial input for the development of the UI while they were taking the Semantic Technology research seminar at the University of Innsbruck.

CHAPTER 6

Towards Knowledge Graphs Validation through Weighted Knowledge Sources

This chapter is published as:

Huaman, E., Tauqeer, A., Fensel, A. (2021). Towards Knowledge Graphs Validation Through Weighted Knowledge Sources. In: Villazón-Terrazas, B., Ortiz-Rodríguez, F., Tiwari, S., Goyal, A., Jabbar, M. (eds) Knowledge Graphs and Semantic Web. KGSWC 2021. Communications in Computer and Information Science, vol 1459. Springer, Cham. https://doi.org/10.1007/978-3-030-91305-2_4 (Huaman et al., 2021)

Abstract

The performance of applications, such as personal assistants and search engines, relies on high-quality knowledge bases, a.k.a. Knowledge Graphs (KGs). To ensure their quality one important task is knowledge validation, which measures the degree to which statements or triples of KGs are semantically correct. KGs inevitably contain incorrect and incomplete statements, which may hinder their adoption in business applications as they are not trustworthy. In this paper, we propose and implement a Validator that computes a confidence score for every triple and instance in KGs. The computed score is based on finding the same instances across different weighted knowledge sources and comparing their features. We evaluate our approach by comparing its results against a baseline validation. Our results suggest that we can validate KGs with an f-measure of at least 75%. Time-wise, the Validator, performed a validation of 2530 instances in 15 minutes approximately. Furthermore, we give insights and directions toward a better architecture to tackle KG validation.

Keywords— knowledge graph validation, knowledge graph curation, knowledge graph assessment.

6.1 Introduction

Over the last decade, creating and especially maintaining knowledge bases have gained attention, and therefore large knowledge bases, also known as knowledge graphs (KGs) (Hogan et al., 2021), have been created, either automatically (e.g. NELL (Carlson et al., 2010)), semi-automatically (e.g. DBpedia (Auer et al., 2007)), or through crowdsourcing (e.g. Freebase (Bollacker et al., 2008)). Today, open (e.g. Wikidata) and proprietary (e.g. Knowledge Vault) KGs provide information about entities like hotels, places, restaurants, and statements about them, e.g. address, phone number, and website. With the increasing use of KGs in personal assistant and search engine applications, the need to ensure that statements or triples in KGs are correct arises (Fensel et al., 2020; Huaman et al., 2020; Paulheim, 2017). For example, Google shows the fact (*Gartenhotel Maria Theresia GmbH, phone, 05223 563130*), which might be wrong because the $phone$ number of the $Gartenhotel$ is *05223 56313*, moreover, there will be cases where the $phone$ number is not up-to-date or is missing (Kärle et al., 2016).

To face this challenge, we developed an approach to validate a KG against different knowledge sources. Our approach involves (1) mapping the different knowledge sources to a common schema (e.g. Schema.org[1]), (2) instance matching that ensures that we are comparing the same entity across the different knowledge sources, (3) confidence measurement, which computes a confidence score for each triple and instance in the KG, and (4) visualization that offers an interface to interact with. Furthermore, we describe use cases where our approach can be used.

There have been a few approaches proposed to validate KGs. In this paper, we review methods, tools, and benchmarks for knowledge validation. We found out that most of them focus on validating knowledge against the Web or Wikipedia. For example, the approaches measure the degree to which a statement (e.g. *Paris is the capital of France*) is true based on the number of occurrences of the statement in sources such as Wikipedia, websites, and/or textual corpora. In addition, to the best of our knowledge, no studies have investigated how to validate KGs by collecting matched instances from other weighted structured knowledge sources.

In this paper, we propose a weighted approach that validates a KG against a set of weighted knowledge sources, which have different weight (or degree of importance) for different application scenarios. For example, users can define the degree of importance of a knowledge source according to the task at hand. We validate a KG by finding the same instances across different knowledge sources, comparing their features, and scoring them. The score ranges from 0 to 1, which indicates the degree to which an instance is semantically correct for the task at hand.

This paper is structured as follows. Section 6.2 presents related state-of-the-art methods, tools, and benchmarks. Section 6.3 describes our validation approach. We evaluate our approach and show its results in Section 6.4. Furthermore, in Section 6.5

[1]https://schema.org/

we list use cases where our approach may be needed. Finally, we conclude with Section 6.6, providing some remarks and future work plans.

6.2 Literature Review

Knowledge Validation (KV), a.k.a. fact checking, is the task of assessing how likely a given fact or statement is true or semantically correct (Gad-Elrab et al., 2019; Lehmann et al., 2012; Rula et al., 2019). There are currently several state-of-the-art methods and tools available that are suitable for KV. One of the prior works on automating this task focuses on analysing trustworthiness factors of web search results (e.g. the trustworthiness of web pages based on topic majority, which computes the number of pages related to a query) (Nakamura et al., 2007). Another approach is proposed by Yin et al. (Yin et al., 2008). Here, the authors define the trustworthiness of a website based on the confidence of facts provided by the website, for instance, they propose an algorithm called *TruthFinder*. Moreover, (Dong et al., 2014) present Knowledge Vault, which is a probabilistic knowledge base that combines information extraction and machine learning techniques to compute the probability that a statement is correct. The computed score is based on knowledge extracted from the Web and corroborative paths found on Freebase. However, the Web can yield noisy data and prior knowledge bases may be incomplete. Therefore, we propose an approach that not only takes into account the user's preferences for weighting knowledge sources, but also complements the existing probabilistic approaches.

We surveyed methods for validating statements in KGs and we distinguish them according to the data used by them, as follows: a) *internal* approaches use the knowledge graph itself as input and b) *external* approaches use external data sources (e.g. DBpedia) as input. In the context of this paper, we only consider the approaches that use external knowledge sources for validating statements.

The external approaches use external sources like the DBpedia source to validate a statement. For instance, there are approaches that use websites information (Dong et al., 2014; Gerber et al., 2015; Speck & Ngomo, 2019), Wikipedia pages (Ercan et al., 2019; Padia et al., 2018; Syed et al., 2018), DBpedia knowledge base (Liu et al., 2015; Rula et al., 2019), and so on. In contrast to other approaches, (Liu et al., 2015) present an early stage approach that uses DBpedia to find out $sameAs$ links, which are followed for retrieving evidence triples in other knowledge sources and (Rula et al., 2019) uses DBpedia to retrieve temporal constraints for a fact. However, (Liu et al., 2015) do not provide an evaluation of the approach to be compared with our approach and (Rula et al., 2019) focus on validating dynamic data, which we do not tackle in the scope of this paper. Furthermore, there are methods that use topic coherence (Aletras & Stevenson, 2013) and information extraction (Speck & Ngomo, 2019) techniques to validate knowledge. Obviously, there is not only one approach or ideal solution to validate KGs. The proposed tools – DeFacto[2], Leopard[3], FactCheck[4], and FacTify[5]– rely on the Web and/or external knowledge sources like Wikipedia.

The current Web-based approaches can effectively validate knowledge that is well disseminated on the Web, e.g. *Albert Einstein's date of birth is March 14, 1879.* Furthermore, the confidence score is based on the number of occurrences of a statement in a corpus (e.g. Wikipedia). Unfortunately these approaches are also prone to spamming (Tan et al., 2014). Therefore, a new approach is necessary to further improve KG validation. In this paper, we propose a KG validation approach, which computes a confidence score for each triple and instance of KGs.

Furthermore, an evaluation of validation approaches is really important, therefore, we also surveyed knowledge validation benchmarks that have been proposed, however, the number of them is currently rather limited. (Vlachos & Riedel, 2014) and (Ercan et al., 2019) released a benchmark consisting of triples extracted from a KG (e.g. Yago) and textual evidences retrieved from a corpus (e.g., Wikipedia). Furthermore. (Thorne et al., 2018) released FEVER[6] that is a dataset containing 185K claims about entities which were verified using Wikipedia articles. Moreover, FactBench[7] (Fact Validation Benchmark) provides a multilingual (i.e. English, German and French) benchmark that describes several relations (e.g. Award, Birth, Death, Foundation Place) of entities.

All benchmarks mentioned above have focused mostly on textual sources, i.e. unstructured information. Therefore, from the best of our knowledge, there is no available benchmark that can be used for validating knowledge graphs via collecting matched instances from other structured knowledge sources.

Last but not least, the reviewed approaches are mostly focused on validating well disseminated knowledge than factual knowledge. Furthermore, benchmarks are built for validating specific tools or to be used during contests like FEVER. Another interesting observation is that Wikipedia is the most frequently used by external approaches (i.e. Wikipedia as textual corpus for finding evidences). Finally, to make future works on knowledge graph validation comparable, it would be useful to have a common selection of benchmarks.

6.3 Approach

In this section, we present the conceptualization of our KG validation approach. First, we give an overview of the knowledge validation process (see Fig. 6.1). Second, we state

[2]https://github.com/DeFacto/DeFacto
[3]https://github.com/dice-group/Leopard
[4]https://github.com/dice-group/FactCheck
[5]http://qweb.cs.aau.dk/factify/
[6]https://github.com/sheffieldnlp/fever-naacl-2018
[7]https://github.com/DeFacto/FactBench

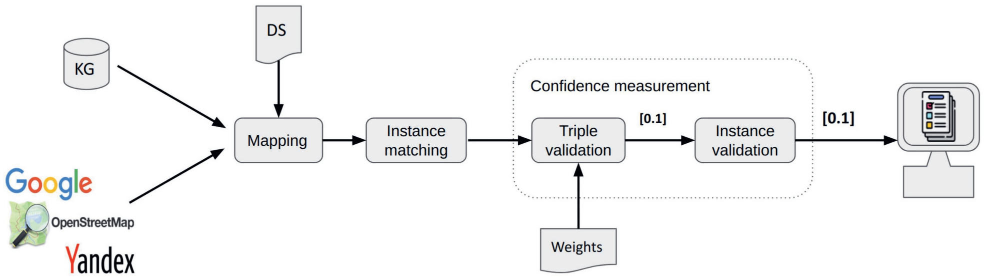


Figure 6.1: Knowledge graph validation process overview.

the input needed for our approach in Section 6.3.1. In Section 6.3.2, we describe the need for a common attribute space between knowledge sources. Then, in Section 6.3.3, we explain the instance matching process. Afterwards, confidence measurement of instances is detailed in Section 6.3.4. Finally, in Section 6.3.5, we describe the output of our implemented approach.

The input to the Validator is a KG, which can be provided via a SPARQL endpoint or an RDF dataset in Turtle[8] format. This input KG is first mapped based on a Domain Specification[9] (DS), which basically defines the mapping of the KG to a common format, e.g., this process may be performed by a domain expert, who defines the types and properties that are relevant to the task at hand or the user's need (Simsek et al., 2020). A DS defines the instance type and properties values to be validated. Internally, the Validator is configured to retrieve data from external sources, which are also mapped to the common format. After the mapping process has been done, the instance matching is used to find the same instances across the KG and the external sources. Then, the confidence measurement process is triggered and the features of same instances are compared with each other. For example, we compare the name value of an instance of the KG against the name value of the same instance in an external source. We repeat this process for every triple of an instance and we compute a triple confidence score, the triple confidence scores are later added to an aggregated confidence score for the instance. The computed scores are normalized according to the weights given to each knowledge source. We consider the quality of the external sources subjective, therefore, we provide a graphical user interface that allows users to weight each knowledge source.

[8]https://www.w3.org/TR/turtle/

[9]Domain Specification are design patterns for annotating data based on Schema.org. This process implies to remove types and properties from Schema.org, or add types and properties defined in an external extension of Schema.org.

6.3.1 Input

At first step, a user is required to provide a KG to be validated. For this, the user has two options, a) to provide a SPARQL endpoint where to fetch the data from or b) to load a dataset in a Turtle format. Moreover, the user is required to select, from a list of DSs, a DS that defines an instance type (e.g., Hotel, Person) and their corresponding properties (e.g., name, address). Internally, the Validator has been set up to fetch data from different external sources (e.g. Wikidata, DBpedia), which were selected based on their domain coverage for the task at hand and their widely use (Färber et al., 2018).

6.3.2 Mapping

Based on the DS defined in the input, the validator maps the input KG and the external sources to a common format, e.g., a telephone number of a hotel can be stored with different property names across the knowledge sources: *phone*, *telephone*, or *phone_number*. The validator provides a basic mapping feature to map the input KG and external data sources to a common attribute space. This step is not trivial. There is a huge number of knowledge sources and their schemas might be constantly changing (Dong & Srivastava, 2015). As a result, schema alignment[10] is one of the major bottlenecks in the mapping process. Therefore, new methods and frameworks to tackle the schema heterogeneity are needed.

6.3.3 Instance Matching

So far, we mapped knowledge sources to a common attribute space. However, a major challenge is to match instances across these knowledge sources. For that, the Validator requests to define at least two or more properties (e.g., *name* and *geo coordinates*) that are to be used for the instance matching process, which is constrained to strict matches on the defined property values. The resulting matched instance is returned to the Validator and processed to measure its confidence.

6.3.4 Confidence Measurement

Computing a confidence value can get complicated as the number of instances and their features can get out of hand quickly. Therefore, a means to automatically validate KGs is desirable. To compute a confidence value for an instance, the confidence value for each of its triples has to be evaluated first.

Triple validation

[10]Schema alignment is the task of determining the correspondences between various schemas.

calculates a confidence score of whether a property value on various external sources matches the property value in the user's KG. For example, the user's KG contains the *Hotel Alpenhof* instance and statements about it; *Hotel Alpenhof's phone is +4352878550* and *Hotel Alpenhof's address is Hintertux 750*. Furthermore, there are other sources, like Google Places, that also contain the *Hotel Alpenhof* instance and assertions about it.

The confidence score of *(Hotel Alpenhof, phone, +4352878550)* triple is computed by comparing the phone property value *+4352878550* against the same property value of the same instance in Google Places. For that, syntactic similarity matching of the attribute values is used. Then the phone property value is compared against a second knowledge source, and so on. Every similarity comparison returns a confidence value that later is added to an aggregated score for the triple.

We define a set of knowledge sources as S, $S = \{s_1, \ldots, s_m\}$, $s_i \in S$ with $1 \leq i \leq m$. The user's KG g consists of a set of instances that are to be validated against the set of knowledge sources S. A knowledge source s_i consists of a set of instances $E = \{e_1, \ldots, e_n\}$, $e_j \in E$ with $1 \leq j \leq n$ and an instance e_j consists of a set of attribute values $P = \{p_1, \ldots, p_M\}$, $p_k \in P$ for $1 \leq k \leq M$.

Furthermore, sim is a similarity function used to compare attribute pair k for two instances. We compute the similarity of an attribute value of two instances a, b. Where a represents an instance in the user's KG g, denoted $g(a)$, and b represents an instance in the knowledge source s_i, denoted $s_i(b)$.

$$triple_{confidence}(a_{p_k}, S, sim) = \sum_{i=1}^{m} sim(g(a_{p_k}), s_i(b_{p_k})) \tag{6.1}$$

Next, users have to set an external weight for each knowledge source s_i, $W = \{\omega_1, \ldots, \omega_m\}$ is a set of weights over the knowledge sources, such as ω_i defines a weight of importance for s_i, $0 \leq i \leq m$, $\omega_i \in W$ with $\omega_i \in [0, 1]$ where 0 is the minimum degree of importance and a value of 1 is the maximum degree. For the sum of weights $w_{sum} = \sum_{i=1}^{m} \omega_i = 1$ has to hold. We compute the weighted triple confidence as follows:

$$triple_{confidence}(a_{p_k}, S, sim, W) = \frac{1}{w_{sum}} \sum_{i=1}^{m} sim(g(a_{p_k}), s_i(b_{p_k}))w_i \tag{6.2}$$

The weighted approach[11] aims to model the different degrees of importance of different knowledge sources. None of the parameters can be taken out of their context, thus a default weight has to be given whenever the user does not set weights for an external source. The Validator assigns an equivalent weight for each source: $\omega_i = \frac{1}{m}$.

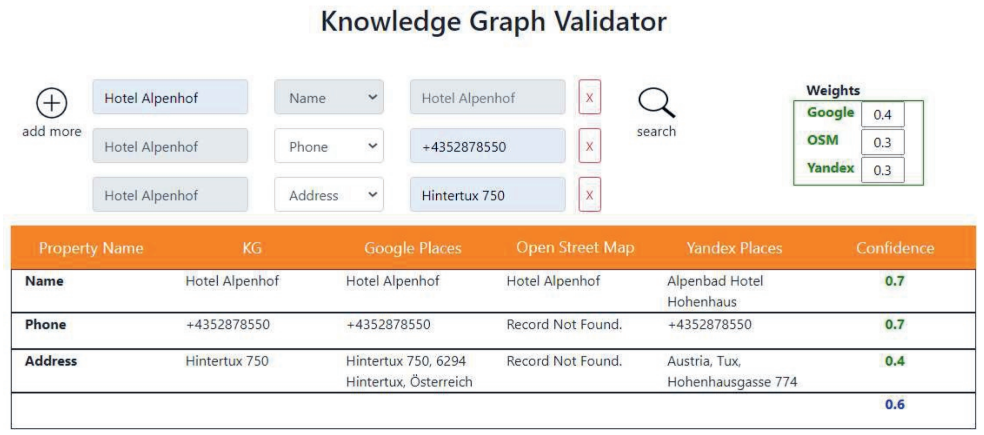
[11]To define weights, a proper quality analysis of the knowledge sources must be carried out (Färber et al., 2018). It may assist users in defining degrees of importance for each knowledge source.

Instance validation

computes the aggregated score from the attribute space of an instance. Given an instance a that consists of a set of attribute values $P = \{p_1, \ldots, p_M\}$, $p_k \in P$ for $1 \leq k \leq M$:

$$instance_{confidence}(a_{p_k}, S, sim, W) = \frac{1}{M}\sum_{k=1}^{M} triple_{confidence}(a_{p_k}, S, sim, W) \qquad (6.3)$$

The instance confidence measures the degree to which an instance is correct based on the triple confidence of each of its attributes. The instance confidence score is compared against a threshold[12] $t \in [0, 1]$. If $instance_{confidence} > t$ indicates its degree of correctness.



Hotel Alpenhof	Name	Hotel Alpenhof	X
Hotel Alpenhof	Phone	+4352878550	X
Hotel Alpenhof	Address	Hintertux 750	X

Google	0.4
OSM	0.3
Yandex	0.3

Property Name	KG	Google Places	Open Street Map	Yandex Places	Confidence
Name	Hotel Alpenhof	Hotel Alpenhof	Hotel Alpenhof	Alpenbad Hotel Hohenhaus	0.7
Phone	+4352878550	+4352878550	Record Not Found.	+4352878550	0.7
Address	Hintertux 750	Hintertux 750, 6294 Hintertux, Österreich	Record Not Found.	Austria, Tux, Hohenhausgasse 774	0.4
					0.6

Figure 6.2: Screenshot of the Validator Web interface.

6.3.5 Output

The computed scores for triples and instances are shown in a graphical user interface, see Fig. 6.2. The interface provides many features: it allows users to select multiple properties (e.g. address, name) to be validated, users can assign weights to external sources, it shows instance information from user's KG and external sources. For example, the Validator shows information of the *Hotel Alpenhof* instance from all sources. It also shows the triple confidence score for each triple, e.g. the triple confidence for the address property is 0.4, because the address value is confirmed only by Google Places.

Tools & Technologies We implemented our approach in the Validator tool[13], which has been implemented in JavaScript[14] for retrieving data remotely, and Bootstrap[15] for the user interface.

[12]The default threshold is defined to 0.5
[13]https://github.com/AmarTauqeer/graph-validation
[14]https://developer.mozilla.org/en-US/docs/Web/JavaScript
[15]https://getbootstrap.com/

6.4 Evaluation

This section describes the evaluation of our approach. The aim of the experiments is to show a qualitative and quantitative analysis of our approach. The setup used for the evaluation is described in Table 6.1.

Table 6.1: Evaluation setup

CPU	RAM	OS
AMD Ryzen 7 pro 4750u (16 Cores)	32GB	Ubuntu 20.04.2 LTS 64-bit

In Section 6.4.1, we compare the Validator's validation result against a baseline. Next, we look into the scalability of the Validator in Section 6.4.2.

6.4.1 Qualitative evaluation

The qualitative evaluation measures the effectiveness of the Validator based on a baseline validation. To do so, first, we describe a dataset to be used on the quality evaluation of our approach, later on we define a setup for the Validator and execute it. Then, we stablish a baseline to compare the result of the Validator.

Hotel dataset.

It was fetched from the Tirol Knowledge Graph[16] (TKG), which contains ~ 15 Billion statements about hotels, places, and more, of the Tirol region. The data inside the TKG are static (e.g name, phone number) and dynamic (e.g. availability of rooms, prices) and are based on Schema.org annotations, which are collected from different sources such as destination management organizations and geographical information systems. We have created a benchmark dataset of 50 hotel instances[17] fetched from the TKG. We randomly selected 50 hotel instances in order to be able to perform a manual validation of their correctness and establish a baseline. The process of creating the Hotel dataset involved manual checking of the correctness of all instances and their attribute values.

Setup and Execution.

First, we set up the Hotel dataset on the Validator. Second, we defined external sources, namely: Google Places[18], OpenStreetMap (OSM)[19], and Yandex Places[20]. Third, we defined the *Hotel* type and *address*, *name*, and *phone* properties that are used for mapping place instances from external sources. Then, for the instance matching process, we set up the *name* and *geo-coordinates* values to search for places within

[16]https://graphdb.sti2.at/sparql
[17]https://github.com/AmarTauqeer/graph-validation/tree/master/data
[18]https://developers.google.com/maps/documentation/places/
[19]https://www.openstreetmap.org/
[20]https://yandex.com/dev/maps/

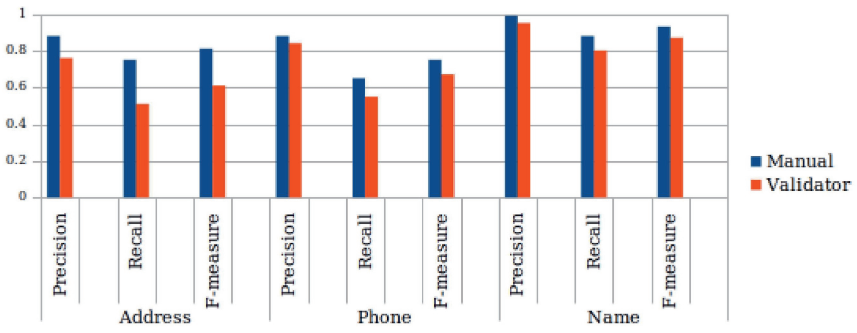


Figure 6.3: Comparison of precision, recall, and f-measure scores over the manual and semi-automatic validation.

a specified area. We use the built-in feature provided by the external sources (e.g. Nearby Search for Google places) to search for an instance with the same name within a specific area. Furthermore, weights for the external sources are equally distributed. Finally, we run the validation task.

Baseline.

In order to evaluate the results of the Validator, a baseline must be established. Given that no prior validation tool addresses exactly the task at hand, we made a manual validation of the Hotel dataset. We computed the precision, recall, and f-measure that a manual validation would achieve (See Fig. 6.3). During this evaluation, the 50 hotel instances are manually searched and compared to the results coming from each of the external knowledge sources: Google Places, OSM, and Yandex Places. The compared attributes are the *address, name*, and *phone*.

Result.

We analyse the result of running the Validator on the Hotel dataset. These results are shown in Fig. 6.3. On one hand, it shows that the Validator performs almost equally similar as the manual evaluation when it comes to *name* and *phone* properties, on the other hand, the Validator does not perform well on the validation of the *address* property. Moreover, the results suggest that we can validate hotel instances with an f-measure of at least 75% on *address*, *name*, and *phone* properties. To interpret the results of our validation run, we choose precision, recall, and f-measure. Given the results of the Validator run, every validated triple result was classified as True Positive, False Positive, True Negative, or False Negative based on the baseline results.

6.4.2 Scalability evaluation

Another challenge of a validation framework is the scalability. In this section, we describe our evaluation approach in terms of scalability of our approach.

Pantheon dataset

It contains manually validated data with 11341 famous biographies (Yu et al., 2016). Pantheon describes information like name, year of birth, place of birth, occupation, and many more. We have selected politician domain and created a dataset of 2530 politician instances. We selected the politician domain because it has the highest number of instances in the Pantheon dataset. Furthermore, we had to convert the Pantheon dataset to Turtle format, for that we used Tarql[21] tool. Last but not least, we selected the politician domain in order to prove the general applicability of our approach in different domains (e.g., Hotel, Person).

Setup and Execution

The setup for validating datasets from different domains changes slightly, for example, defining the external sources where to fetch the data from. First, we set up the Pantheon dataset on the Validator. Then, we defined Wikidata and DBpedia as external sources and we distributed equivalent weights for them. Moreover, we defined the *person* type and *name* and *year of birth* properties for mapping politicians from the external sources. Moreover, we set up the *name* and *year of birth* for the instance matching process. Finally, we execute the validation task.

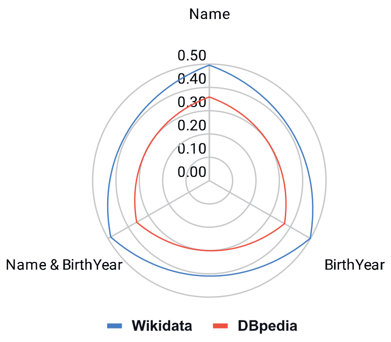


Figure 6.4: The recall score results of the validation of politician instances.

Result.

We validated 2530 politician instances by using the Validator, which compares and computes a confidence score for each triple and instance. To execute this task the Validator required $\sim$15 minutes approximately on a CPU described on Table 6.1. Results are presented in Fig. 6.4. On one hand, it shows that Wikidata outperforms DBpedia on validated properties, on the other hand, it shows lower recall scores, by the Validator, on both sources, e.g. the overall recall scores are 0.36% (DBpedia) and 0.49% (Wikidata).

[21]https://tarql.github.io/

Furthermore, the Validator gets lower recall on DBpedia and Wikidata sources due to two reasons. First, DBpedia contains the validated politician instances, however many of them are classified in DBpedia as *agent* type and not as politician (e.g., *Juan Carlos I*[22]). Second, the Wikidata query service raised timeout errors when querying data, so we decided to fetch the maximum allowed number of politician instances from Wikidata and stored them locally. We fetched 45000 out of 670810 politicians.

6.5 Use Cases

Our approach, as described in Section 6.3, aims to validate KGs by finding the same instances across different knowledge sources and comparing their features. Later on, based on the compared features our approach computes a confidence score for each triple and instance, the confidence score ranges from 0 to 1 and indicates the degree to which an instance is correct. Our approach may be used in a variety of use cases, we list some of the cases where the approach can be used:

- To validate the semantic correctness of a triple, e.g., to validate if the phone number of a hotel is the correct based on different sources.
- To link instances between knowledge sources, e.g. linking an instance of the user's KG with the matched instance in Wikidata.
- To find out incorrect data on different knowledge sources. For instance, suppose that the owner of a hotel wants to validate whether the information of his or her hotel provided by an external source are up-to-date.
- To validate static data, for example, to check whether the addresses of hotels are still valid given a period of time.

There are more possible use cases where our validation approach is applicable. Here, we presented some of them to give an idea about how useful and necessary is to have a validated KG (i.e. a correct and reliable KG).

6.6 Conclusion and Future Work

In this paper, we presented the conceptualization of a new KG validation approach and a first prototypical implementation thereof. Our approach measures the degree to which every instance in a KG is semantically correct. It evaluates the correctness of instances based on external sources. Experiments were conducted on two datasets. The results confirm its effectiveness and are promising great potential. In future work, we will improve our approach and overcome its limitations. Here, we give a short overview of them:

[22]https://dbpedia.org/page/Juan_Carlos_I

- **Assessment** of knowledge sources. Finding the most suitable knowledge source for validating a KG is challenging (Färber et al., 2018). Therefore, it is desirable to implement a quality assessment mechanism for assessing external sources. It may assist users in defining degrees of importance for each knowledge source.
- **Automation** of the setting process. It is desirable to allow users to create a semi-automatic **mapping** (or schema alignment (Dong & Srivastava, 2015)) between their KG and the external sources, e.g. the heterogeneous scheme of OSM has caused low performance of the Validator (see Section 6.4.1).
- **Cost-sensitive methods**. The current version of the Validator relies on proprietary services like Google, which can lead to high costs when validating large KGs. Therefore, it is important to evaluate the cost-effectiveness of knowledge sources.
- **Dynamic data** is fast-changing data that also needs to be validated, e.g. the price of a hotel room. The scope of this paper only comprises the validation of static data.
- **Scalability** is a critical point when we want to validate KGs. KGs are very large semantic networks that can contain billions of statements.

Above, we pointed out some future research directions and improvements that one can implement on the development of future validation tools.

Acknowledgments This work has been partially funded by the project WordLiftNG within the Eureka, Eurostars Programme of the European Union (grant agreement number 877857 with the Austrian Research Promotion Agency (FFG)) and the industrial research project MindLab[23]. We would like to thank Prof. Dr. Dieter Fensel for his insightful comments regarding the definition of the overall validation approach.

[23]https://mindlab.ai/

Bibliography

Aghaei, S. Question Answering over Knowledge Graphs. In *5th Doctoral Consortium at Rules and Reasoning (RuleML+RR)*. CEUR Workshop Proceedings (CEUR-WS.org), 2021. https://ceur-ws.org/Vol-2956/paper35.pdf

Aghaei, S., Angele, K., & Fensel, A. Building Knowledge Subgraphs in Question Answering over Knowledge Graphs. In *Web Engineering*. Cham: Springer International Publishing, 2022, 237–251. ISBN: 978-3-031-09917-5. https://doi.org/10.1007/978-3-031-09917-5_16

Aghaei, S., Angele, K., Huaman, E., Bushati, G., Schiestl, M., & Fensel, A. (2022b). Interactive Search on the Web: The Story So Far. *Information*, *13*(7). https://doi.org/10.3390/info13070324

Aghaei, S., Raad, E., & Fensel, A. (2022c). Question Answering Over Knowledge Graphs: A Case Study in Tourism. *IEEE Access*, *10*, 69788–69801. https://doi.org/10.1109/ACCESS.2022.3187178

Ahmed, N., Michelin, R. A., Xue, W., Ruj, S., Malaney, R., Kanhere, S. S., Seneviratne, A., Hu, W., Janicke, H., & Jha, S. K. (2020). A Survey of COVID-19 Contact Tracing Apps. *IEEE Access*, *8*, 134577–134601. https://doi.org/10.1109/ACCESS.2020.3010226

Ahmetaj, S., David, R., Ortiz, M., Polleres, A., Shehu, B., & Šimkus, M. Reasoning about Explanations for Non-validation in SHACL. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*. 2021, November, 12–21. https://doi.org/10.24963/kr.2021/2

Ahmetaj, S., David, R., Polleres, A., & Šimkus, M. Repairing SHACL Constraint Violations Using Answer Set Programming. In *The Semantic Web – ISWC 2022*. Cham: Springer International Publishing, 2022, 375–391. ISBN: 978-3-031-19433-7. https://doi.org/10.1007/978-3-031-19433-7_22

Aletras, N., & Stevenson, M. Evaluating Topic Coherence Using Distributional Semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS2013), Potsdam, Germany, March 19-22, 2013*. The Association for Computer Linguistics, 2013, 13–22. https://aclanthology.org/W13-0102.pdf

Algarni, M. A. Contract Lifecycle Management: Processes and Benefits. In *Innovative and Agile Contracting for Digital Transformation and Industry 4.0*. IGI Global, 2021, pp. 62–85. https://doi.org/10.4018/978-1-7998-4501-0.ch004

Allemang, D., Hendler, J. A., & Gandon, F. (2020). *Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL*. ACM Press. https://doi.org/10.1145/3382097

Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A. P., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J., Richardson, J., Ringwald, M., Rubin, G., & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nature genetics*, *25*(1), 25–29. https://doi.org/10.1038/75556

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. G. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC2007), 2nd Asian Semantic Web Conference, (ASWC2007), Busan, Korea, November 11-15, 2007. 4825*. Lecture Notes in Computer Science. Springer, 2007, 722–735. https://doi.org/10.1007/978-3-540-76298-0_52

Baeza-Yates, R. (2018). Bias on the Web. *Communications of the ACM*, *61*(6), 54–61. https://doi.org/10.1145/3209581

Barati, M., Adu-Duodu, K., Rana, O., Aujla, G. S., & Ranjan, R. (2023). Compliance Checking of Cloud Providers: Design and Implementation. *Distributed Ledger Technologies: Research and Practice*. https://doi.org/10.1145/3585538

Barati, M., & Rana, O. (2022). Tracking GDPR Compliance in Cloud-Based Service Delivery. *IEEE Transactions on Services Computing*, *15*(3), 1498–1511. https://doi.org/10.1109/TSC.2020.2999559

Beckett, D. (2014). Rdf 1.1 n-triples [Available at: https://www.w3.org/TR/n-triples/, Accessed on 01 August 2023].

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., & Carothers, G. (2014). RDF 1.1 Turtle [Available at: https://core.ac.uk/download/pdf/70283847.pdf, Accessed on 01 August 2023]. *World Wide Web Consortium*, 18–31.

Bennett, M. (2013). The financial industry business ontology: Best practice for big data. *Journal of Banking Regulation*, *14*(3-4), 255–268. https://doi.org/10.1057/jbr.2013.13

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities [Available at: https://www.jstor.org/stable/pdf/26059207.pdf, Accessed on 01 August 2023]. *ScientificAmerican.com*.

BFO Discussion Group. (2007). Basic Formal Ontology (BFO) 2.0 [Available online: https://github.com/bfo-ontology/BFO/wiki, Accessed on 10 June 2023].

Bibri, S. E., & Krogstie, J. (2020). The emerging data–driven smart city and its innovative applied solutions for sustainability: the cases of London and Barcelona. *Energy Informatics*, *3*(1), 1–42. https://doi.org/10.1186/s42162-020-00108-6

Bless, C., Dötlinger, L., Kaltschmid, M., Reiter, M., Kurteva, A., Roa-Valverde, A. J., & Fensel, A. Raising Awareness of Data Sharing Consent Through Knowledge Graph Visualisation. In *SEMANTiCS Conference, Further with Knowledge Graphs. 53*. 2021, 44 –57. https://doi.org/10.3233/SSW210034

Boesveldt, S., Postma, E. M., Boak, D., Welge-Luessen, A., Schöpf, V., Mainland, J. D., Martens, J., Ngai, J., & Duffy, V. B. (2017). Anosmia—A Clinical Review. *Chemical Senses*, *42*(7), 513–523. https://doi.org/10.1093/chemse/bjx025

Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge. In *Proceedings of the 2008 ACM International Conference on Management of Data (SIGMOD2008), Vancouver, Canada, June 09 - 12, 2008*. ACM, 2008, 1247–1250. https://doi.org/10.1145/1376616.1376746

Borst, W. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse* (Doctoral dissertation). University of Twente. Netherlands, Centre for Telematics and Information Technology (CTIT). https://ris.utwente.nl/ws/portalfiles/portal/6036651/t0000004.pdf

Botts, M., & Robin, A. (2007). OpenGIS® Sensor Model Language (SensorML) Implementation Specification. Version 1.0. 0. https://doi.org/10.25607/OBP-646

Breitfuss, A., Errou, K., Kurteva, A., & Fensel, A. (2021). Representing emotions with knowledge graphs for movie recommendations. *Future Generation*

Computer Systems, *125*, 715–725. https://doi.org/10.1016/j.future.2021.06.001

Brickley, D., Guha, R. V., & Layman, A. (1998). *Resource description framework (RDF) schema specification* (tech. rep.) [Available at: https://www.immagic.com/eLibrary/ARCHIVES/SUPRSDED/W3C/W990303P.pdf, Accessed on 07 August 2023]. Technical report, W3C, 1999. W3C Proposed Recommendation.

Cai, T., Wu, Y., Lin, H., & Cai, Y. Blockchain-empowered big data sharing for internet of things. In *Research Anthology on Convergence of Blockchain, Internet of Things, and Security*. IGI Global, 2023, pp. 278–290. https://doi.org/10.4018/978-1-6684-7132-6.ch017

Cambronero, M.-E., Llana, L., & Pace, G. J. Timed Contract Compliance Under Event Timing Uncertainty. In *JURIX*. 2017, 33–38. https://doi.org/10.3233/978-1-61499-838-9-33

Cao, Q. H., Madhusudan, G., Farahbakhsh, R., & Crespi, N. Usage control for data handling in smart cities. In *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2015, 1–6. https://doi.org/10.1109/GLOCOM.2015.7417270

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., & Mitchell, T. M. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI2010), Atlanta, Georgia, July 11 - 15, 2010*. AAAI Press, 2010, 1306–1313. https://doi.org/10.5555/2898607.2898816

Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J., & Fischer, A. (2019). Introduction to neural network based approaches for question answering over knowledge graphs. *arXiv preprint arXiv:1907.09361*. https://doi.org/10.48550/arXiv.1907.09361

Chhetri, T. R. Improving Decision Making Using Semantic Web Technologies. In *The Semantic Web: European Semantic Web conference (ESWC) Satellite Events*. Cham: Springer International Publishing, 2021, 165–175. https://doi.org/10.1007/978-3-030-80418-3_29

Chhetri, T. R., Kurteva, A., Adigun, J. G., & Fensel, A. (2022a). Knowledge Graph Based Hard Drive Failure Prediction. *Sensors*, *22*(3), 985. https://doi.org/10.3390/s22030985

Chhetri, T. R., Kurteva, A., DeLong, R. J., Hilscher, R., Korte, K., & Fensel, A. (2022b). Data Protection by Design Tool for Automated GDPR Compliance Verification Based on Semantically Modeled Informed Consent. *Sensors*, *22*(7). https://doi.org/10.3390/s22072763

Chieu, T. C., Nguyen, T., Maradugu, S., & Kwok, T. An enterprise electronic contract management system based on service-oriented architecture. In

IEEE International Conference on Services Computing (SCC 2007). IEEE. 2007, 613–620. https://doi.org/10.1109/SCC.2007.25

Cohen, J., Wakefield, C. E., & Laing, D. G. (2016). Smell and taste disorders resulting from cancer and chemotherapy. *Current Pharmaceutical Design*, *22*(15), 2253–2263. https://doi.org/10.2174/13816128226661602161 50812

Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W. D., Le Phuoc, D., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., . . . Taylor, K. (2012). The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, *17*, 25–32. https://doi.org/10.1016/j.websem.2012.05.003

Corman, J., Reutter, J. L., & Savković, O. Semantics and Validation of Recursive SHACL. In *The Semantic Web – ISWC 2018*. Cham: Springer International Publishing, 2018, 318–336. https://doi.org/10.1007/978-3-030-00671-6_19

Cox, S. (2022). Time Ontology in OWL [Available online: https://www.w3.org/TR/owl-time/, Accessed on 13 April 2022].

Croy, I., Nordin, S., & Hummel, T. (2014). Olfactory Disorders and Quality of Life—An Updated Review. *Chemical Senses*, *39*(3), 185–194. https://doi.org/10.1093/chemse/bjt072

DATA, M. O. S. (1995). Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal L*, *281*(23/11), 0031–0050. https://eur-lex.europa.eu/eli/dir/1995/46/oj

de Cesare, S., & Geerts, G. L. Toward a perdurantist ontology of contracts. In *International Conference on Advanced Information Systems Engineering*. Springer. 2012, 85–96. https://doi.org/10.1007/978-3-642-31069-0_7

De Lauretis, L. From monolithic architecture to microservices architecture. In *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE. 2019, 93–96. https://doi.org/10.1109/ISSREW.2019.00050

Doe, S. (2018). Practical Privacy: Report from the GDPR World. *Legal Information Management*, *18*(2), 76–79. https://doi.org/10.1017/S1472669618000178

Domingue, J., Fensel, D., & Hendler, J. A. (2011). Introduction to the Semantic Web Technologies. In *The Handbook of Semantic Web Technologies* (pp. 1–41). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-92913-0_1

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., & Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining (KDD2014), New York, USA, August 24 - 27, 2014*. ACM, 2014, 601–610. https://doi.org/10.1145/2623330.2623623

Dong, X. L., & Srivastava, D. (2015). *Big Data Integration*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00578ED1V01Y201404DTM040

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, 195–216. https://doi.org/10.1007/978-3-319-67425-4_12

Du, H., Le, Z., Wang, H., Chen, Y., & Yu, J. (2022). COKG-QA: Multi-hop Question Answering over COVID-19 Knowledge Graphs. *Data Intelligence*, *4*(3), 471–492. https://doi.org/10.1162/dint_a_00154

Ducq, Y., & Chen, D. How to measure interoperability: Concept and approach. In *IEEE Int. Technol. Manag. Conf., ICE*. Lisbonne, Portugal: Institute of Electrical and Electronics Engineers Inc., 2008. https://hal.science/hal-03173489

EDM Council. (2020). The Financial Industry Business Ontology (FIBO) [Available online: https://spec.edmcouncil.org/fibo/, Accessed on 11 January 2022].

Eid, M., Liscano, R., & el Saddik, A. A Universal Ontology for Sensor Networks Data. In *2007 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*. 2007, 59–62. https://doi.org/10.1109/CIMSA.2007.4362539

Eiter, T., Ianni, G., & Krennwallner, T. Answer Set Programming: A Primer. In *Reasoning Web. Semantic Technologies for Information Systems. 5689*. Lecture Notes in Computer Science. Springer, 2009, 40–110. https://doi.org/10.1007/978-3-642-03754-2_2

Ercan, G., Elbassuoni, S., & Hose, K. Retrieving Textual Evidence for Knowledge Graph Facts. In *Proceedings of the 16th European Semantic Web Conference (ESWC 2019), Portorož, Slovenia, June 2-6, 2019. 11503*. Lecture Notes in Computer Science. Springer, 2019, 52–67. https://doi.org/10.1007/978-3-030-21348-0_4

European Data Protection Board. (2022). European Data Protection Board. [Available online: https://edpb.europa.eu/edpb_en, Accessed on 26 July 2022].

Färber, M., Bartscherer, F., Menne, C., & Rettinger, A. (2018). Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web*, *9*(1), 77–129. https://doi.org/10.3233/SW-170275

Fatema, K., Hadziselimovic, E., Pandit, H. J., Debruyne, C., Lewis, D., & O'Sullivan, D. (2017). Compliance through Informed Consent: Semantic Based Consent Permission and Data Management Model. *Privacy and the Semantic Web - Policy and Technology workshop co-located with ISWC 2017*. http://www.tara.tcd.ie/bitstream/handle/2262/82659/88248c5b669175f267069c3319d9ac2d3e84.pdf?sequence=1

Fecher, B., Friesike, S., & Hebing, M. (2015). What drives academic data sharing? *PloS one*, *10*(2), e0118053. https://doi.org/10.1371/journal.pone.0118053

Federal Information Processing Standards Publication (FIPS), PUB. (2022). 197. Advanced Encryption Standard (AES), National Institute of Standards and Technology, US Department of Commerce [Available online: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf, Accessed on 20 July 2022].

Fensel, D. (2001). *Ontologies - a silver bullet for knowledge management and electronic commerce*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-09083-1

Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J., & Wahler, A. (2020). *Knowledge Graphs*. Springer. https://doi.org/10.1007/978-3-030-37439-6

Ferrari, V. (2018). EU Blockchain Observatory and Forum Workshop on GDPR, Data Policy and Compliance. *Social Science Research Network (SSRN) Electronic Journal*. https://doi.org/10.2139/ssrn.3247494

Freire, N., & de Valk, S. Automated interpretability of linked data ontologies: an evaluation within the cultural heritage domain. In *IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, 3072–3079. https://doi.org/10.1109/BigData47090.2019.9005491

Fries, Martin AND Paal, Boris P. (Ed.). (2019). *Smart Contracts* (1st ed.). Mohr Siebeck. http://www.jstor.org/stable/j.ctvn96h9r

Gad-Elrab, M. H., Stepanova, D., Urbani, J., & Weikum, G. Tracy: Tracing Facts over Knowledge Graphs and Text. In *Proceedings of the 19th World Wide Web Conference (WWW2019), San Francisco, USA, May 13-17, 2019*. ACM, 2019, 3516–3520. https://doi.org/10.1145/3308558.3314126

Garg, N., & Yadav, P. (2014). Comparison of asymmetric algorithms in cryptography. *Journal of Computer Science and Mobile Computing (IJCSMC)*, *3*(4), 1190–1196. https://doi.org/https://www.ijcsmc.com/docs/papers/April2014/V3I4201499a73.pdf

Gayo, J. E. L., Prud'hommeaux, E., Boneva, I., & Kontokostas, D. (2017). *Validating RDF Data*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00786ED1V01Y201707WBE016

GDPR. (2018a). GDPR (Art. 24) [Available online: https://gdpr.eu/article-24-responsibility-of-the-data-controller/, Accessed on 08 August 2023].

GDPR. (2018b). GDPR (Art. 28) [Available online: https://gdpr.eu/article-28-processor/, Accessed on 08 August 2023].

GDPR. (2018c). GDPR (Art. 32) [Available online: https://gdpr.eu/article-32-security-of-processing/, Accessed on 08 August 2023].

GDPR. (2018d). GDPR (Art. 4) [Available online: https://gdpr.eu/article-4-definitions/, Accessed on 08 August 2023].

GDPR. (2018e). GDPR (Art. 5) [Available online: https://gdpr.eu/article-5-how-to-process-personal-data/, Accessed on 08 August 2023].

GDPR. (2018f). GDPR (Art. 6) [Available online: https://gdpr.eu/article-6-how-to-process-personal-data-legally/, Accessed on 08 August 2023].

GDPR. (2018g). GDPR (Art. 7) [Available online: https://gdpr.eu/article-7-how-to-get-consent-to-collect-personal-data/, Accessed on 08 August 2023].

GDPR. (2018h). GDPR (Art. 83) [Available online: https://gdpr.eu/article-83-conditions-for-imposing-administrative-fines/, Accessed on 08 August 2023].

GDPR. (2018i). GDPR Rec.42 [Available online: https://gdpr-info.eu/recitals/no-42/, Accessed on 08 August 2023].

GDPR. (2018j). GDPR Rec.43 [Available online: https://gdpr-info.eu/recitals/no-43/, Accessed on 08 August 2023].

GDPR. (2018k). GDPR Rec.44 [Available online: https://gdpr-info.eu/recitals/no-44/, Accessed on 08 August 2023].

GDPR. (2018l). GDPR Rec.82 [Available online: https://gdpr-info.eu/recitals/no-82/, Accessed on 08 August 2023].

Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2019). Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, *19*(1), 27–82. https://doi.org/10.1017/S1471068418000054

Genome Sciences, Institute. (2022). Human Disease Ontology [Available online: https://disease-ontology.org/, Accessed on 10 July 2023].

Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngomo, A. N., & Speck, R. (2015). DeFacto - Temporal and multilingual Deep Fact Validation. *Journal of Web Semantics*, *35*, 85–101. https://doi.org/10.1016/j.websem.2015.08.001

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, *5*(2), 199–220. https://doi.org/https://doi.org/10.1006/knac.1993.1008

Gruninger, M., & Lee, J. (2002). Introduction. *Communications of the ACM*, *45*(2), 39–41. https://doi.org/10.1145/503124.503146

Guo, L., Liu, Q., Shi, K., Gao, Y., Luo, J., & Chen, J. (2021). A Blockchain-Driven Electronic Contract Management System for Commodity Procurement in Electronic Power Industry. *IEEE Access*, *9*, 9473–9480. https://doi.org/10.1109/ACCESS.2021.3049562

Guédria, W., Naudet, Y., & Chen, D. (2015). Maturity model for enterprise interoperability. *Enterprise Information Systems*, *9*(1), 1–28. https://doi.org/10.1080/17517575.2013.805246

Habib, H., Li, M., Young, E., & Cranor, L. "Okay, Whatever": An Evaluation of Cookie Consent Interfaces. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. https://doi.org/10.1145/3491102.3501985

Haller, A., Janowicz, K., Cox, S., Lefrançois, M., Taylor, K., Le-Phuoc, D., Lieberman, J., García-Castro, R., Atkinson, R., & Stadler, C. (2019). The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. *Semantic Web*, *10*, 9–32. https://doi.org/10.3233/SW-180320

Hashem, I. A. T., Chang, V., Anuar, N. B., Adewole, K., Yaqoob, I., Gani, A., Ahmed, E., & Chiroma, H. (2016). The role of big data in smart city. *International Journal of Information Management*, *36*(5), 748–758. https://doi.org/10.1016/j.ijinfomgt.2016.05.002

Hassan, F. U., Le, T., & Le, C. (2023). Automated Approach for Digitalizing Scope of Work Requirements to Support Contract Management. *Journal of Construction Engineering and Management*, *149*(4), 04023005. https://doi.org/10.1061/JCEMD4.COENG-12528

Havur, G., Steyskal, S., Panasiuk, O., Fensel, A., Mireles, V., Pellegrini, T., Thurner, T., Polleres, A., & Kirrane, S. DALICC: A framework for publishing and consuming data assets legally. In *SEMANTICS Posters & Demos*. 2018. https://doi.org/https://ceur-ws.org/Vol-2198/paper_114.pdf.

Hogan, A. (2020). Web Ontology Language. In *The Web of Data* (pp. 185–322). Springer International Publishing. https://doi.org/10.1007/978-3-030-51580-5_5

Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). Knowledge Graphs. *Synthesis Lectures on Data, Semantics, and Knowledge*, *12*(2), 1–257. https://doi.org/10.1145/3447772

Hooi, Y. K., Hassan, M. F., & Shariff, A. M. Ontology evaluation — A criteria selection framework. In *2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC)*. 2015, 298–303. https://doi.org/10.1109/ISMSC.2015.7594069

Huaman, E., Kärle, E., & Fensel, D. (2020). Knowledge Graph Validation. *CoRR, abs/2005.01389*. https://doi.org/10.48550/arXiv.2005.01389

Huaman, E., Tauqeer, A., & Fensel, A. Towards Knowledge Graphs Validation Through Weighted Knowledge Sources. In *The Knowledge Graphs and Semantic Web*. Cham: Springer International Publishing, 2021, 47–60. ISBN: 978-3-030-91305-2. https://doi.org/https://doi.org/10.1007/978-3-030-91305-2_4.

Huang, J., Kong, L., Wang, J., Chen, G., Gao, J., Huang, G., & Khan, M. K. (2023). Secure Data Sharing over Vehicular Networks Based on Multi-Sharding Blockchain. *ACM Trans. Sen. Netw.* https://doi.org/10.1145/3579035

Hummel, T., Landis, B. N., & Hüttenbrink, K.-B. (2011). Smell and taste disorders. *GMS current topics in otorhinolaryngology, head and neck surgery, 10*. https://doi.org/10.3205/cto000077

Hunhevicz, J. J., Motie, M., & Hall, D. M. (2022). Digital building twins and blockchain for performance-based (smart) contracts. *Automation in Construction, 133*, 103981. https://doi.org/10.1016/j.autcon.2021.103981

Iannella, R. (2004). The Open Digital Rights Language: XML for Digital Rights Management. *Information Security Technical Report, 9*(3), 47 –55. https://doi.org/10.1016/S1363-4127(04)00031-7

Irwin, K., Yu, T., & Winsborough, W. H. On the modeling and analysis of obligations. In *13th ACM conference on Computer and communications security*. 2006, 134–143. https://doi.org/10.1145/1180405.1180423

Ízdemir, F., & Ídemiş Ízger, Z. Rivest-Shamir-Adleman Algorithm. In *Partially Homomorphic Encryption*. Springer, 2021, pp. 37–41. https://doi.org/10.1007/978-3-030-87629-6_3

Janowicz, K., Haller, A., Cox, S. J., Le Phuoc, D., & Lefrançois, M. (2019). SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics, 56*, 1–10. https://doi.org/10.1016/j.websem.2018.06.003

Jones, M., Bradley, J., & Sakimura, N. (2010). JSON Web Tokens (JWT). [Available online: https://jwt.io/, Accessed on 20 June 2022].

Jusic, A. (2020). Dealing with Tensions Between the Blockchain and the GDPR. *The LegalTech Book: The Legal Technology Handbook for Investors, Entrepreneurs and FinTech Visionaries*, 83–86. https://doi.org/10.1002/9781119708063.ch22

Jusic, A. (2022). Privacy between Regulation and Technology: GDPR and the Blockchain [https://ssrn.com/abstract=4049371]. *International University of Sarajevo (IUS) Law Journal*, *1*(1), 47–59.

Jussen, I., Schweihoff, J., Dahms, V., Möller, F., & Otto, B. Data Sharing Fundamentals: Definition and Characteristics. In *Proceedings of the 56th Hawaii International Conference on System Sciences*. 2023. http://dx.doi.org/10.24251/HICSS.2023.452

Kabilan, V., & Johannesson, P. Semantic Representation of Contract Knowledge Using Multi Tier Ontology. In *First International Conference on Semantic Web and Databases*. SWDB'03. Berlin, Germany: CEUR-WS.org, 2003, 378–397. https://dl.acm.org/doi/abs/10.5555/2889905.2889930

Kaffee, L.-A., Piscopo, A., Vougiouklis, P., Simperl, E., Carr, L., & Pintscher, L. A Glimpse into Babel: An Analysis of Multilinguality in Wikidata. In *Proceedings of the 13th International Symposium on Open Collaboration*. OpenSym '17. Galway, Ireland: Association for Computing Machinery, 2017. ISBN: 9781450351874. https://doi.org/10.1145/3125433.3125465

Kalaycı, E. G., Grangel González, I., Lösch, F., Xiao, G., ul Mehdi, A., Kharlamov, E., & Calvanese, D. Semantic Integration of Bosch Manufacturing Data Using Virtual Knowledge Graphs. In *The Semantic Web – ISWC 2020*. Cham: Springer International Publishing, 2020, 464–481. ISBN: 978-3-030-62466-8. https://doi.org/10.1007/978-3-030-62466-8_29

Kärle, E., Fensel, A., Toma, I., & Fensel, D. Why Are There More Hotels in Tyrol than in Austria? Analyzing Schema.org Usage in the Hotel Domain. In *Proceedings of the Conference on Information and Communication Technologies in Tourism (ENTER2016), Bilbao, Spain, February 2-5, 2016*. Springer, 2016, 99–112. https://doi.org/10.1007/978-3-319-28231-2_8

Keller, A., & Ludwig, H. (2003). The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, *11*, 57–81. https://doi.org/10.1023/A:1022445108617

Kim, H. H., Kim, B., Joo, S., Shin, S.-Y., Cha, H. S., & Park, Y. R. (2019). Why Do Data Users Say Health Care Data Are Difficult to Use? A Cross-Sectional Survey Study. *J Med Internet Res*, *21*(8), e14126. https://doi.org/10.2196/14126

Kirrane, S., Fernández, J. D., Bonatti, P., Milosevic, U., Polleres, A., & Wenning, R. (2020). The SPECIAL-K Personal Data Processing Transparency and Compliance Platform. *arXiv:2001.09461*. https://doi.org/10.48550/arXiv.2001.09461

Knoke, F., & Nwankwo, I. (2022). Practitioner's corner managing data protection compliance through maturity models: A primer. *European Data Protection Law Review*, *8*(4). https://doi.org/10.21552/edpl/2022/4/14

Knublauch, H., & Kontokostas, D. (2017). *Shapes constraint language (SHACL)* (tech. rep.). W3C. https://www.w3.org/TR/shacl/

Kuo, A. M.-H. (2011). Opportunities and Challenges of Cloud Computing to Improve Health Care Services. *J Med Internet Res*, *13*(3), e67. https://doi.org/10.2196/jmir.1867

Kurteva, A. Implementing Informed Consent with Knowledge Graphs. In *The Semantic Web: European Semantic Web conference (ESWC) Satellite Events*. Cham: Springer International Publishing, 2021, 155–164. https://doi.org/10.1007/978-3-030-80418-3_28

Kurteva, A. (2023). Making Sense of Consent with Knowledge Graphs [Available online: https://www.researchgate.net/profile/Anelia-Kurteva/publication/367052478_Making_Sense_of_Consent_with_Knowledge_Graphs/links/63bfd0d5a99551743e60a486/Making-Sense-of-Consent-with-Knowledge-Graphs.pdf, Accessed on 4 June 2023].

Kurteva, A., Chhetri, T. R., Pandit, H. J., & Fensel, A. (2021). Consent Through the Lens of Semantics: State of the Art Survey and Best Practices. *Semantic Web journal*. https://doi.org/10.3233/SW-210438

Kurteva, A., Chhetri, T. R., Tauqeer, A., Hilscher, R., Fensel, A., Nagorny, K., Correia, A., Zilverberg, A., Schestakov, S., Funke, T., & Demidova, E. (2023). The smashHitCore Ontology for GDPR-Compliant Sensor Data Sharing in Smart Cities. *Sensors*, *23*(13). https://doi.org/10.3390/s23136188

Köhler, S., Gargano, M., Matentzoglu, N., Carmody, L. C., Lewis-Smith, D., Vasilevsky, N. A., Danis, D., Balagura, G., Baynam, G., Brower, A. M., Callahan, T. J., Chute, C. G., Est, J. L., Galer, P. D., Ganesan, S., Griese, M., Haimel, M., Pazmandi, J., Hanauer, M., . . . Robinson, P. N. (2020). The Human Phenotype Ontology in 2021. *Nucleic Acids Research*, *49*(D1), D1207–D1217. https://doi.org/10.1093/nar/gkaa1043

Lakehal, A., Alti, A., & Roose, P. (2019). A semantic event based framework for complex situations modeling and identification in smart environments. *International Journal of Advanced Computer Research*, *9*(43), 212–221. https://doi.org/10.19101/IJACR.PID33

Landis, B. N., Welge-Luessen, A., Brämerson, A., Bende, M., Mueller, C. A., Nordin, S., & Hummel, T. (2009). "Taste Strips"–a rapid, lateralized, gustatory bedside identification test based on impregnated filter papers. *Journal of neurology*, *256*(2), 242–248. https://doi.org/10.1007/s00415-009-0088-y

Laudisio, A., Navarini, L., Margiotta, D. P. E., Fontana, D. O., Chiarella, I., Spitaleri, D., Bandinelli, S., Gemma, A., Ferrucci, L., & Incalzi, R. A. (2019). The association of olfactory dysfunction, frailty, and mortality is mediated by inflammation: results from the InCHIANTI Study. *Journal of immunology research, 2019*. https://doi.org/10.1155/2019/3128231

Lawful basis for processing [Available at https://ico.org.uk/media/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/lawful-basis-for-processing-1-0.pdf, Accessed on 09 November 2021]. (2018).

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., & Zhao, J. (2013). Prov-o: The prov ontology [Available online: https://www.w3.org/TR/prov-o/]. *World Wide Web Consortium, 30*.

Lehmann, J., Gerber, D., Morsey, M., & Ngomo, A. N. DeFacto - Deep Fact Validation. In *Proceedings of the 11th International Semantic Web Conference (ISWC2012), Boston, MA, USA, November 11-15, 2012. 7649*. Lecture Notes in Computer Science. Springer, 2012, 312–327. https://doi.org/10.1007/978-3-642-35176-1_20

Li, H., Yu, L., & He, W. (2019). The Impact of GDPR on Global Technology Development. *Journal of Global Information Technology Management*, *22*(1), 1–6. https://doi.org/10.1080/1097198X.2019.1569186

Li, M., & Samavi, R. DSAP: Data Sharing Agreement Privacy Ontology. In *SWAT4LS*. 2018. https://doi.org/10.6084/m9.figshare.7322420

Licht, T. R. (2001). The IEEE 1451.4 proposed standard. *IEEE Instrumentation & Measurement Magazine*, *4*(1), 12–18. https://doi.org/10.1109/5289.911168

Lin, Y., Harris, M. R., Manion, F. J., Eisenhauer, E., Zhao, B., Shi, W., Karnovsky, A., & He, Y. (2014). Development of a BFO-based informed consent ontology (ICO). *Bioinformatics*, *1327*, 84–86. https://ceur-ws.org/Vol-1327/icbo2014_paper_54.pdf

Liu, C., & Yang, S. (2022). Using text mining to establish knowledge graph from accident/incident reports in risk assessment. *Expert Systems with Applications*, *207*, 117991. https://doi.org/10.1016/j.eswa.2022.117991

Liu, L., Tsai, W.-T., Bhuiyan, M. Z. A., Peng, H., & Liu, M. (2022). Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Generation Computer Systems*, *128*, 158–166. https://doi.org/10.1016/j.future.2021.08.023

Liu, S., d'Aquin, M., & Motta, E. Towards Linked Data Fact Validation through Measuring Consensus. In *Proceedings of the 2nd Workshop on Linked Data Quality co-located with 12th Extended Semantic Web Conference*

(ESWC2015), Portoro, Slovenia, June 1, 2015. 1376. CEUR Workshop proceedings. CEUR-WS.org, 2015. https://ceur-ws.org/Vol-1376/LDQ2015_paper_04.pdf

Liu, Y., & Huang, J. (2019). Legal Creation of Smart Contracts and the Legal Effects. *Journal of Physics: Conference Series*, *1345*, 042033. https://doi.org/10.1088/1742-6596/1345/4/042033

Longo, A., Zappatore, M., & Bochicchio, M. A. Service Level Aware - Contract Management. In *2015 IEEE International Conference on Services Computing*. 2015, 499–506. https://doi.org/10.1109/SCC.2015.74

Loukil, F., Ghedira, C., Boukadi, K., & Benharkat, A. (2018). LIoPY: A Legal Compliant Ontology to Preserve Privacy for the Internet of Things. *IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, *02*, 701–706. https://doi.org/10.1109/COMPSAC.2018.10322

Mamadolimova, A., Ambiah, N., & Lukose, D. Modeling Islamic finance knowledge for contract compliance in Islamic banking. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer. 2011, 346–355. https://doi.org/10.1007/978-3-642-23854-3_37

Mangini, V., Tal, I., & Moldovan, A.-N. An empirical study on the impact of GDPR and right to be forgotten-organisations and users perspective. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*. 2020, 1–9. https://doi.org/10.1145/3407023.3407080

Matentzoglu, N., Malone, J., Mungall, C., & Stevens, R. (2018). MIRO: guidelines for minimum information for the reporting of an ontology. *Journal of Biomedical Semantics*, *9*(1), 1–13. https://doi.org/10.1186/s13326-017-0172-7

Matthias, G. (2019). THE IMPACT OF THE GDPR ON THIRD-PARTY CONTRACTS IN THE CLOUD SERVICE INDUSTRY (Tilburg University, 2019). [Available online: http://arno.uvt.nl/show.cgi?fid=149355, Accessed on 20 July 2022].

McBride, B. (2023). RDF Schema 1.1 [Available online: https://www.w3.org/TR/rdf-schema/, Accessed on 08 August 2023].

McCray, A. T. (2003). An upper-level ontology for the biomedical domain. *Comparative and functional genomics*, *4*(1), 80–84. https://doi.org/10.1002/cfg.255

Mellal, N., Guerram, T., & Bouhalassa, F. (2021). An approach for automatic ontology enrichment from texts. *Informatica*, *45*(1). https://doi.org/10.31449/inf.v45i1.2586

Miller, E. (2001). An introduction to the resource description framework. *Journal of Library Administration*, *34*, 245–255. https://doi.org/10.1300/J111v34n03_04

Mondschein, C. F., & Monda, C. (2019). The EU's General Data Protection Regulation (GDPR) in a Research Context. In *Fundamentals of Clinical Data Science* (pp. 55–71). Springer International Publishing. https://doi.org/10.1007/978-3-319-99713-1_5

Murtazina, M. S., & Avdeenko, T. (2019). An ontology-based approach to support for requirements traceability in agile development. *Procedia Computer Science*, *150*, 628–635. https://doi.org/10.1016/j.procs.2019.02.044

MyData Global and KIRAHub. (2022). H3C event: Built for people - human-centric solutions for the built environment [Available online: https://old www.mydata.org/h3c-event-built-for-people-6-april-2022/, Accessed on 30 March 2023].

Nakamura, S., Konishi, S., Jatowt, A., Ohshima, H., Kondo, H., Tezuka, T., Oyama, S., & Tanaka, K. Trustworthiness Analysis of Web Search Results. In *Proceedings of the 11th European Conference on Research and Advanced Technology for Digital Libraries (ECDL2007), Budapest, Hungary, September 16-21, 2007. 4675*. Lecture Notes in Computer Science. Springer, 2007, 38–49. https://doi.org/10.1007/978-3-540-74851-9_4

Nefzi, H., Farah, M., Farah, I. R., & Solaiman, B. A semi-automatic mapping selection in the ontology alignment process. In *KEOD 2014 : 6th International conference on knowledge engineering and ontology development.* Rome, Italy, 2014. https://hal.archives-ouvertes.fr/hal-01186365

Nicholson, D. N., & Greene, C. S. (2020). Constructing knowledge graphs and their biomedical applications. *Computational and Structural Biotechnology Journal*, *18*, 1414–1428. https://doi.org/10.1016/j.csbj.2020.05.017

Niles, I., & Pease, A. Towards a standard upper ontology. In *International conference on Formal Ontology in Information Systems. 2001*. 2001, 2–9. https://doi.org/10.1145/505168.505170

Noy, N., & McGuinness, D. L. Ontology Development 101: A Guide to Creating Your First Ontology. In *Technical report KSL-01-05 and SMI-2001-0880.* Stanford Knowledge Systems Laboratory and Stanford Medical Informatics, 2001. https://api.semanticscholar.org/CorpusID:500106

Object Management Group. (2019). Languages, countries, and codes (LCC) Version 1.1 beta 2 [Available at https://www.omg.org/spec/LCC/1.1/Beta1/PDF, Accessed on 26 December 2021].

Oleszkiewicz, A, Kunkel, F, Larsson, M., & Hummel, T. (2020). Consequences of undetected olfactory loss for human chemosensory communication and well-being. *Philosophical Transactions of the Royal Society B*, *375*(1800), 20190265. https://doi.org/10.1098/rstb.2019.0265

Oleszkiewicz, A, Schriever, V., Croy, I, Hähner, A, & Hummel, T. (2019). Updated Sniffin'Sticks normative data based on an extended sample of 9139 subjects. *European Archives of Oto-Rhino-Laryngology*, *276*(3), 719–728. https://doi.org/10.1007/s00405-018-5248-1

Otte, J. N., Beverley, J., & Ruttenberg, A. (2022). Bfo: Basic Formal Ontology. *Applied ontology*, *17*(1), 17–43. https://doi.org/10.3233/ao-220262

Padia, A., Ferraro, F., & Finin, T. (2018). SURFACE: Semantically Rich Fact Validation with Explanations. *CoRR*, *abs/1810.13223*. https://doi.org/10.48550/arXiv.1810.13223

Palmirani, M., Martoni, M., Rossi, A., Cesare, B., & Livio, R. Pronto: Privacy ontology for legal compliance. In *Proceedings of the European Conference on e-Government, ECEG*. Springer International Publishing, 2018, 142–151. https://doi.org/10.1007/978-3-642-31069-0_7

Pan, J. Z. (2009). Resource Description Framework. In *Handbook on Ontologies* (pp. 71–90). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-92673-3_3

Pandit, H. J., Debruyne, C., O'Sullivan, D., & Lewis, D. (2019a). GConsent - A Consent Ontology Based on the GDPR. *The Semantic Web. European Semantic Web Conference (ESWC) 2019. Lecture Notes in Computer Science*, *11503*, 270–282. https://doi.org/10.1007/978-3-030-21348-0_18

Pandit, H. J., Polleres, A., Bos, B., Brennan, R., Bruegger, B., Ekaputra, F. J., Fernández, J. D., Hamed, R. G., Kiesling, E., Lizar, M., Schlehahn, E., Steyskal, S., & Wenning, R. Creating a vocabulary for data privacy: The first-year report of data privacy vocabularies and controls community group (DPVCG). In *On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21–25, 2019, proceedings*. Springer. 2019, 714–730. https://w3c.github.io/dpv/dpv/

Pandit, H. J., O'Sullivan, D., & Lewis, D. Towards Knowledge-Based Systems for GDPR Compliance. In *International Semantic Web Conference (ISWC), Monterey, California, USA*. 2019. https://doi.org/10.5281/zenodo.3246477

Pantlin, N., Wiseman, C., & Everett, M. (2018). Supply chain arrangements: The ABC to GDPR compliance—A spotlight on emerging market practice in supplier contracts in light of the GDPR. *Computer Law & Security Review*, *34*(4), 881–885. https://doi.org/10.1016/j.clsr.2018.06.009

Park, J., Cho, Y., Lee, H., Choo, J., & Choi, E. Knowledge Graph-based Question Answering with Electronic Health Records. In *Proceedings of the 6th Machine Learning for Healthcare Conference. 149*. Proceedings of Machine

Learning Research. PMLR, 2021, 36–53. https://proceedings.mlr.press/v149/park21a.html

Parma, V., Ohla, K., Veldhuizen, M. G., Niv, M. Y., Kelly, C. E., Bakke, A. J., Cooper, K. W., Bouysset, C., Pirastu, N., Dibattista, M., Kaur, R., Liuzza, M. T., Pepino, M. Y., Schöpf, V., Pereda-Loth, V., Olsson, S. B., Gerkin, R. C., Rohlfs Domínguez, P., Albayay, J., . . . Hayes, J. E. (2020). More than smell—COVID-19 is associated with severe impairment of smell, taste, and chemesthesis. *Chemical senses*, *45*(7), 609–622. https://doi.org/10.1093/chemse/bjaa041

Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, *8*(3), 489–508. https://doi.org/10.3233/SW-160218

Pellegrini, T., Havur, G., Steyskal, S., Panasiuk, O., Fensel, A., Mireles, V., Thurner, T., Polleres, A., Kirrane, S., & Schönhofer, A. DALICC: A License Management Framework for Digital Assets. In *Internationales Rechtsinformatik Symposion (IRIS). 10.* 2019. https://www.researchgate.net/publication/332414717_DALICC_A_LICENSE_MANAGEMENT_FRAMEWORK_FOR_DIGITAL_ASSETS

Pellegrini, T., Mireles, V., Steyskal, S., Panasiuk, O., Fensel, A., & Kirrane, S. (2018). Automated Rights Clearance Using Semantic Web Technologies: The DALICC Framework. In *The Semantic Applications: Methodology, Technology, Corporate Use* (pp. 203–218). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-55433-3_14

Pérez, J., Arenas, M., & Gutierrez, C. (2009). Semantics and Complexity of SPARQL. *ACM Trans. Database Syst.*, *34*(3). https://doi.org/10.1145/1567274.1567278

Perrin, O., & Godart, C. An approach to implement contracts as trusted intermediaries. In *First IEEE International Workshop on Electronic Contracting.* 2004, 71–78. https://doi.org/10.1109/WEC.2004.1319511

Petasis, G., Karkaletsis, V., Paliouras, G., Krithara, A., & Zavitsanos, E. (2011). Ontology Population and Enrichment: State of the Art. In *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution: Bridging the Semantic Gap* (134–166). Springer-Verlag. https://doi.org/10.1007/978-3-642-20795-2_6

Petrova, G. G., Tuzovsky, A. F., & Aksenova, N. V. (2017). Application of the Financial Industry Business Ontology (FIBO) for development of a financial organization ontology. *Journal of Physics: Conference Series*, *803*, 012116. https://doi.org/10.1088/1742-6596/803/1/012116

Poveda-Villalón, M., Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2014). Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation.

International Journal on Semantic Web and Information Systems (IJSWIS), *10*(2), 7–34. https://doi.org/10.4018/ijswis.2014040102

Pradhan, N., & Singh, A. (2021). Smart contracts for automated control system in Blockchain based smart cities. *Journal of Ambient Intelligence and Smart Environments*, *13*, 1–15. https://doi.org/10.3233/AIS-210601

Python Software Foundation. (2021). Unit testing framework. [Available online: https://docs.python.org/3/library/unittest.html, Accessed on 16 June 2022].

Rabbani, K., Lissandrini, M., & Hose, K. SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption. In *Companion Proceedings of the Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, 260–263. ISBN: 9781450391306. https://doi.org/10.1145/3487553.3524253

Ramayanti, D., Ayumi, V., Noprisson, H., Ratnasari, A., Handriani, I., Utami, M., & Putra, E. D. Tuberculosis Ontology Generation and Enrichment Based Text Mining. In *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*. IEEE. 2020, 429–434. https://doi.org/10.1109/ICITSI50517.2020.9264922

Rasmusen, S., Widauer, S., Penz, M., Nako, P., Kurteva, A., Roa-Valverde, A. J., & Fensel, A. (2022). Raising Consent Awareness with Gamification and Knowledge Graphs: An Automotive Use Case. *International Journal on Semantic Web and Information Systems (IJSWIS)*, *18*. https://doi.org/10.4018/IJSWIS.300820

Rasmusen, S. C. (2022). Increasing Trust and Engagement in the Age of GDPR: A Digital Contracting Tool Supported by Knowledge Graphs [Available online: https://ulb-dok.uibk.ac.at/ulbtirolhs/content/titleinfo/8171087, Accessed on 4 June 2023].

Röscheisen, M., Baldonado, M., Chang, K., Gravano, L., Ketchpel, S., & Paepcke, A. (1998). The Stanford InfoBus and its service layers: Augmenting the internet with higher-level information management protocols. In *Digital Libraries in Computer Science: The MeDoc Approach* (pp. 213–230). Springer Berlin Heidelberg. https://doi.org/10.1007/BFb0052526

Rula, A., Palmonari, M., Rubinacci, S., Ngomo, A. N., Lehmann, J., Maurino, A., & Esteves, D. (2019). TISCO: Temporal scoping of facts. *Journal of Web Semantics*, *54*, 72–86. https://doi.org/10.1016/j.websem.2018.09.002

Russomanno, D., Kothari, C., & Thomas, O. A. Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In *International Conference on Artificial Intelligence*. *2*. 2005. https://www.researchgate.net/publication/220834903_Building_a_Sensor_Ontology_A_Practical_Approach_Leveraging_ISO_and_OGC_Models

Sabou, M. (2016). An Introduction to Semantic Web Technologies. In *Semantic Web Technologies for Intelligent Engineering Applications* (pp. 53–81). Springer International Publishing. https://doi.org/10.1007/978-3-319-41490-4_3

Sabou, M., Ekaputra, F. J., & Biffl, S. (2017). emantic Web Technologies for Data Integration in Multi-Disciplinary Engineering. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects* (pp. 301–329). Springer International Publishing. https://doi.org/10.1007/978-3-319-56345-9_12

Salvadores, M., Horridge, M., Alexander, P. R., Fergerson, R. W., Musen, M. A., & Noy, N. F. Using SPARQL to Query BioPortal Ontologies and Metadata. In *The Semantic Web – ISWC 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 180–195. ISBN: 978-3-642-35173-0. https://www.researchgate.net/publication/262365535_Using_SPARQL_to_query_bioportal_ontologies_and_metadata

Savelyev, A. (2017). Contract law 2.0: 'Smart' contracts as the beginning of the end of classic contract law. *Information Communications Technology Law*, *26*, 1–19. https://doi.org/10.1080/13600834.2017.1301036

Savithramma, R. M., Ashwini, B. P., & Sumathi, R. Smart Mobility Implementation in Smart Cities: A Comprehensive Review on State-of-art Technologies. In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2022, 10–17. https://doi.org/10.1109/ICSSIT53264.2022.9716288

Scheuermann, R. H., Ceusters, W., & Smith, B. (2009). Toward an ontological treatment of disease and diagnosis. *Summit on translational bioinformatics*, *2009*, 116. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3041577/

Schmidt, R. N., House, M., & Rodriguez, O. O. (2019). Journey into Dnv Hospital Accreditation and the Implementation of an Electronic Contract Management System (Ecms). *Journal of Business and Educational Leadership*, *9*(1), 67–75. https://asbbs.org/files/2019/JBEL_Vol9_Fall_2019.pdf#page=67

Selent, D. (2010). Advanced encryption standard. *Rivier Academic Journal*, *6*(2), 1–14. https://www2.rivier.edu/journal/roaj-fall-2010/j455-selent-aes.pdf

Semantha, F. H., Azam, S., Shanmugam, B., & Yeo, K. C. (2023). PbDinEHR: A Novel Privacy by Design Developed Framework Using Distributed Data Storage and Sharing for Secure and Scalable Electronic Health Records Management. *Journal of Sensor and Actuator Networks*, *12*(2), 36. https://doi.org/10.3390/jsan12020036

Semantic Agreement [Available online: https://joinup.ec.europa.eu/taxonomy/term/http_e_f_fdata_ceuropa_ceu_fdr8_fSemanticAgreement, Accessed on 20 July 2022]. (2022).

Sermet, Y., & Demir, I. (2021). A Semantic Web Framework for Automated Smart Assistants: A Case Study for Public Health. *Big Data and Cognitive Computing*, *5*(4). https://doi.org/10.3390/bdcc5040057

SHACL Working Group. (2017). SHapes And Constraints Language (SHACL). [Available online: https://www.w3.org/2001/sw/wiki/SHACL, Accessed on 16 June 2022].

Sharma, S. Fact-finding knowledge-aware search engine. In *Data Management, Analytics and Innovation*. Springer, 2022, pp. 225–235. https://doi.org/10.1007/978-981-16-2937-2_17

Shearer, R. D., Motik, B., & Horrocks, I. Hermit: A highly-efficient OWL reasoner. In *Owled. 432*. 2008, 91. https://www.cs.ox.ac.uk/boris.motik/pubs/smh08HermiT.pdf

Sheng, M., Li, A., Bu, Y., Dong, J., Zhang, Y., Li, X., Li, C., & Xing, C. DSQA: A Domain Specific QA System for Smart Health Based on Knowledge Graph. In *Web Information Systems and Applications: 17th International Conference, WISA 2020, Guangzhou, China, September 23–25, 2020, proceedings*. Guangzhou, China: Springer-Verlag, 2020, 215–222. ISBN: 978-3-030-60028-0. https://doi.org/10.1007/978-3-030-60029-7_20

Simić, S., Marković, M., & Gostojić, S. Smart Contract and Blockchain Based Contract Management System. In *7th Conference on the Engineering of Computer Based Systems*. ECBS 2021. Novi Sad, Serbia: Association for Computing Machinery, 2021. ISBN: 9781450390576. https://doi.org/10.1145/3459960.3459975

Simsek, U., Angele, K., Kärle, E., Panasiuk, O., & Fensel, D. Domain-Specific Customization of Schema.org Based on SHACL. In *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, proceedings, Part II. 12507*. Lecture Notes in Computer Science. Springer, 2020, 585–600. https://doi.org/10.1007/978-3-030-62466-8_36

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, *5*(2), 51–53. https://doi.org/10.1016/j.websem.2007.03.004

Sirma Group. (2019). Ontotext GraphDB. [Available online: https://www.ontotext.com/products/graphdb/, Accessed on 01 June 2022].

SmartBear. (2018). Swagger API Documentation. [Available online: https://swagger.io/solutions/api-documentation/, Accessed on 01 June 2022].

smashHit consortium. (2022a). smashHit Concept (white paper) [White paper is part of the smashHit project deliverable D2.2 smashHit Methodology (final)]. https://doi.org/10.5281/zenodo.7870318

smashHit consortium. (2022b). smashHit Data Owner user guide [User guide part of the smashHit project deliverable D2.2 smashHit Methodology (final)]. https://doi.org/10.5281/zenodo.7870389

smashHit consortium. (2022c). smashHit Semantic Model Technical Essay [Technical essay is part of the smashHit project deliverable D2.2 smashHit Methodology (final)]. https://doi.org/10.5281/zenodo.7869284

SPARQL Working Group. (2013). Resource Description Framework (RDF) Sparql Query. [Available online: https://www.w3.org/TR/rdf-sparql-query/, Accessed on 01 June 2022].

SPARQL Working Group. (2008). SPARQL Query Language for RDF [Available online: https://www.w3.org/TR/rdf-sparql-query/, Accessed on 2 February 2022].

Speck, R., & Ngomo, A. N. (2019). Leopard - A baseline approach to attribute prediction and validation for knowledge graph population. *Journal of Web Semantics*, *55*, 102–107. https://doi.org/10.1016/j.websem.2018.12.006

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., & Lindström, N. (2020). JSON-LD 1.1. *W3C Recommendation, Jul.* https://hal.science/hal-02141614v1/file/mozilla.pdf

Stevenson, G., Knox, S., Dobson, S., & Nixon, P. Ontonym: a collection of upper ontologies for developing pervasive systems. In *The Workshop on Context, Information, and Ontologies at the 6th European Semantic Web conference (ESWC)*. 2009. https://doi.org/10.1145/1552262.1552271

Stott, A. (2014). Open Data For Economic Growth (Transport & ICT Global Practice). Washington, DC: The World Bank [Available online: https://www.worldbank.org/content/dam/Worldbank/document/Open-Data-for-Economic-Growth.pdf, Accessed on 10 June 2023].

Strano, M., Molina-Jimenez, C., & Shrivastava, S. Implementing a Rule-Based Contract Compliance Checker. In *Software Services for e-Business and e-Society*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, 96–111. ISBN: 978-3-642-04280-5. https://doi.org/10.1007/978-3-642-04280-5_9

Studer, R., Benjamins, V., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, *25*(1), 161–197. https://doi.org/10.1016/S0169-023X(97)00056-6

Subramanya, S., & Yi, B. (2006). Digital signatures. *IEEE Potentials*, *25*(2), 5–8. https://doi.org/10.1109/MP.2006.1649003

Susha, I., Rukanova, B., Zuiderwijk, A., Gil-Garcia, J. R., & Gasco Hernandez, M. (2023). Achieving voluntary data sharing in cross sector partnerships: Three partnership models. *Information and Organization*, *33*(1), 100448. https://doi.org/10.1016/j.infoandorg.2023.100448

Swathi, G., Hussain, S. M., Kanakam, P., & Suryanarayana, D. (2016). SPARQL for semantic information retrieval from RDF knowledge base. *International Journal of Engineering Trends and Technology (IJETT)*, (7), 351–354. https://doi.org/10.14445/22315381/IJETT-V41P264

Syed, Z. H., Röder, M., & Ngomo, A. N. FactCheck: Validating RDF Triples Using Textual Evidence. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, (CIKM2018), Torino, Italy, October 22-26, 2018*. ACM, 2018, 1599–1602. https://doi.org/10.1145/3269206.3269308

Taelman, R., Vander Sande, M., & Verborgh, R. GraphQL-LD: linked data querying with GraphQL. In *ISWC2018, the 17th International Semantic Web Conference*. 2018, 1–4. https://comunica.github.io/Article-ISWC2018-Demo-GraphQlLD/

Taleka, M., Makkithaya, K., & V.G., N. (2023). A trusted IoT data sharing and secure oracle based access for agricultural production risk management. *Computers and Electronics in Agriculture*, *204*, 107544. https://doi.org/10.1016/j.compag.2022.107544

Tan, C. H., Agichtein, E., Ipeirotis, P., & Gabrilovich, E. Trust, but verify: predicting contribution quality for knowledge base construction and curation. In *The seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*. ACM, 2014, 553–562. https://doi.org/10.1145/2556195.2556227

Tauqeer, A. (2021). Contract Compliance Verification GitHub Repository. [Available online: https://github.com/AmarTauqeer/Contract/tree/master/backend/, Accessed on 01 June 2022].

Tauqeer, A. (2022). Github [Available online: https://github.com/AmarTauqeer/Smell-taste-disorder-question-answering-interface, Accessed on 15 July 2023].

Tauqeer, A., Hammid, I., Aghaei, S., Parvin, P., Postma, E. M., & Fensel, A. (2023). Smell and taste disorders knowledge graph: Answering questions using health data. *Expert Systems with Applications*, *234*, 121049. https://doi.org/10.1016/j.eswa.2023.121049

Tauqeer, A., Kurteva, A., Chhetri, T. R., Ahmeti, A., & Fensel, A. (2022). Automated GDPR Contract Compliance Verification Using Knowledge Graphs. *Information*, *13*(10). https://doi.org/10.3390/info13100447

The BPR4GDPR project. (2019). Compliance Ontology, Deliverable D3.1 [Available at https://www.bpr4gdpr.eu/wp-content/uploads/2019/06/D3.1-Compliance-Ontology-1.0.pdf, Accessed on 10 September 2021].

The smashHit project. (2021). Public Report D1.3 Public Innovation Concept. https://www.smashhit.eu/wp-content/uploads/2021/03/smashHit_D1.3_Public_Innovation_Concept_v100.pdf

The smashHit project. (2020a). The smashHit EU H2020 Project. [Available online: https://smashhit.eu/, Accessed on 20 July 2022].

The smashHit project. (2022). The smashHitCore Ontology [Available at https://github.com/smashhiteu/smashhiteu.github.io/tree/main/ontology, Accessed on 05 March 2023].

The smashHit project. (2020b). UC1 - Insurance Services [Available online: https://smashhit.eu/d6-5-demonstrator-of-services-using-integrated-cpp-and-insurance-data/, Accessed on 9 July 2022].

The smashHit project. (2020c). UC2 - Smart City Services [Available online: https://smashhit.eu/d7-5-demonstrator-of-services-using-integrated-traffic-smart-city-and-cpp-data/, Accessed on 9 July 2022].

Thorne, J., Vlachos, A., Christodoulopoulos, C., & Mittal, A. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT2018), New Orleans, USA, June 1-6, 2018*. Association for Computational Linguistics, 2018, 809–819. https://doi.org/10.18653/v1/N18-1074

Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., & Cimiano, P. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*. 2012, 639–648. https://doi.org/10.1145/2187836.2187923

Urovi, V., Jaiman, V., Angerer, A., & Dumontier, M. (2022). LUCE: A blockchain-based data sharing platform for monitoring data License accoUntability and CompliancE. *Blockchain: Research and Applications*, *3*(4), 100102. https://doi.org/10.1016/j.bcra.2022.100102

Uschold, M., & King, M. Towards a Methodology for Building Ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*. 1995. https://www.aiai.ed.ac.uk/project/oplan/documents/1995/95-ont-ijcai95-ont-method.pdf

Van Rossum, G., Warsaw, B., & Coghlan, N. (2001). PEP 8–style guide for python code. *Python. org, 1565*. http://cnl.sogang.ac.kr/cnlab/lectures/programming/python/PEP8_Style_Guide.pdf

Vlachos, A., & Riedel, S. Fact Checking: Task definition and dataset construction. In *Proceedings of the Workshop on Language Technologies and Computational Social Science (ACL2014), Baltimore, USA, June 26, 2014*. Association for Computational Linguistics, 2014, 18–22. https://doi.org/10.3115/v1/W14-2508

Voronova, O. Development of contract management system for network companies under economy digitalization. In *E3S Web of Conferences. 164*. EDP Sciences. 2020, 09018. https://doi.org/10.1051/e3sconf/202016409018

Voss, W. G. (2021). Data Protection Issues for Smart Contracts. *Smart Contracts: Technological, Business and Legal Perspectives, https://www.bloomsburycollections.com/book/smart-contracts-technological-business-and-legal-perspectives*. https://doi.org/10.5040/9781509937059.ch-004

W3C. (2004a). OWL 2 Web Ontology Language Document Overview (Second Edition) [Available online: https://www.w3.org/TR/owl2-overview/, Accessed on 08 August 2023].

W3C. (2004b). OWL Web Ontology Language (overview) [Available online: https://www.w3.org/TR/2004/REC-owl-features-20040210/, Accessed on 08 August 2023].

W3C. (2004c). OWL Web Ontology Language (reference) [Available online: https://www.w3.org/TR/owl-ref/, Accessed on 21 December 2022].

W3C. (2013). SPARQL 1.1 Protocol [Available online: https://www.w3.org/TR/sparql11-protocol/, Accessed on 08 August 2023].

W3C Community Groups. (2012). Schema.org [Available online: https://schema.org, Accessed on 9 July 2022].

Wang, Z., & Guan, S. (2023). A blockchain-based traceable and secure data-sharing scheme. *PeerJ Computer Science*, *9*, e1337. https://doi.org/10.7717/peerj-cs.1337

World Wide Web Consortium and others. (2014). Data catalog vocabulary (DCAT) [Available online: https://www.w3.org/TR/rdf-sparql-query/, Accessed on 20 February 2022].

Xu, J., Zhao, Y., Chen, H., & Deng, W. (2023). ABC-GSPBFT: PBFT with grouping score mechanism and optimized consensus process for flight operation data-sharing. *Information Sciences*, *624*, 110–127. https://doi.org/10.1016/j.ins.2022.12.068

Yin, X., Han, J., & Yu, P. S. (2008). Truth Discovery with Multiple Conflicting Information Providers on the Web. *IEEE Trans. Knowl. Data Eng.*, *20*(6), 796–808. https://doi.org/10.1109/TKDE.2007.190745

Yu, A. Z., Ronen, S., Hu, K., Lu, T., & Hidalgo, C. A. (2016). Pantheon 1.0, a manually verified dataset of globally famous biographies. *Scientific data*, *3*(1), 1–16. https://doi.org/10.1038/sdata.2015.75

Zaeem, R. N., & Barber, K. S. (2020). The Effect of the GDPR on Privacy Policies: Recent Progress and Future Promise. *ACM Trans. Manage. Inf. Syst.*, *12*(1). https://doi.org/10.1145/3389685

Zafar, A. A., Saif, S., Khan, M., Iqbal, J., Akhunzada, A., Wadood, A., Al-Mogren, A., & Alamri, A. (2018). Taxonomy of Factors Causing Integration Failure during Global Software Development. *IEEE Access*, *6*, 22228–22239. https://doi.org/10.1109/ACCESS.2017.2782843

Zaveri, A, Rula, A, Maurino, A, Pietrobon, R, Lehmann, J, & Auer, S. (2012). Quality assessment methodologies for linked open data. a systematic literature review and conceptual framework. *Semantic Web Journal*, *7*(1), 63–93. https://doi.org/10.3233/SW-150175

Zhou, C., Guan, R., Zhao, C., Chai, G., Wang, L., & Han, X. A Chinese Medical Question Answering System Based on Knowledge Graph. In *2021 IEEE 15th International Conference on Big Data Science and Engineering (BigDataSE)*. IEEE. 2021, 28–33. https://doi.org/10.1109/BigDataSE53435.2021.00014

Zou, J., Wang, Y., & Lin, K.-J. A Formal Service Contract Model for Accountable SaaS and Cloud Services. In *2010 IEEE International Conference on Services Computing*. 2010, 73–80. https://doi.org/10.1109/SCC.2010.85

"Breach of Contract". (2022). Breach of Contract [Available online: http://jec.unm.edu/education/online-training/contract-law-tutorial/breach-of-contract, Accessed on 10 July 2022].

"Contractual necessity". (2019). The contractual necessity basis for processing personal data in the context of online services. [Available online: https://edpb.europa.eu/sites/default/files/consultation/edpb_draft_guidelines-art_6-1-b-final_public_consultation_version_en.pdf, Accessed on 25 July 2022].

"Docker". (2013). Docker. [Available online: https://www.docker.com/, Accessed on 8 June 2022].

"DPV-GDPR". (2022). DPV-GDPR: GDPR Extension for DPV [Available online: https://w3c.github.io/dpv/legal/eu/gdpr/, Accessed on 2 March 2023].

"European Parliament and Council". (2016). European Parliament and Council. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and REPEALING DIRective 95/46/EC (General Data Protection Regulation). Official Journal of the European Union, L119. 2016. [Available online: https://eur-lex.europa.eu/eli/reg/2016/679/oj, Accessed on 05 March 2022].

"European Public Licence". (2017). The European Union Public Licence [Available online: https://ec.europa.eu/isa2/solutions/european-union-public-licence-eupl_en/].

"Flask-Apispec". (2018). Flask-Apispec: Auto-Documenting REST APIs for Flask. [Available online: https://flask-apispec.readthedocs.io/en/latest/, Accessed on 8 June 2022].

"Flask-apscheduler". (2022). Flask-apscheduler. [Available online: https://viniciuschiele.github.io/flask-apscheduler/, Accessed on 01 June 2022].

"Flask-RESTful". (2021). Flask-RESTful. [Available online: https://flask-restful.readthedocs.io/en/latest/, Accessed on 8 June 2022].

"Flask-SQLAlchemy". (2022). Flask-SQLAlchemy. [Available online: https://flask-sqlalchemy.palletsprojects.com/en/2.x/, Accessed on 8 June 2022].

"Flask". (2021). Flask. [Available online: https://flask.palletsprojects.com/en/2.0.x/, Accessed on 8 June 2022].

"GDPR". (2018). General Data Protection Regulation (GDPR). [Available online: https://gdpr.eu/what-is-gdpr/, Accessed on 20 July 2022].

"License Clearance Tool". (2021). License Clearance Tool [Available online: https://wiki.ni4os.eu/index.php/License_Clearance_Tool_-_Description_and_Documentation].

"Marshmallow". (2013-2023). Marshmallow: Simplified Object Serialisation. [Available online: https://marshmallow.readthedocs.io/en/stable/index.html, Accessed on 4 April 2022].

"NIST". (2022). National Institute of Standards and Technology (NIST). [Available online: https://www.nist.gov/, Accessed on 26 July 2022].

"PyCryptodome". (2014). PyCryptodome. [Available online: https://www.pycryptodome.org/en/latest/src/introduction.html, Accessed on 8 June 2022].

"PyJWT". (2015). PyJWT. [Available online: https://pyjwt.readthedocs.io/en/stable/, Accessed on 8 June 2022].

"Python". (2021). Python. [Available online: https://www.python.org/, Accessed on 8 June 2022].

"SDM". (2020). Conference of the Independent Data Protection Supervisory Authorities of the Federation and the Länder. The Standard Data Protection Model—A Method for Data Protection Advising and Controlling on the Basis of Uniform Protection Goals, Version 2.0b. 2020. Available online: https://www.datenschutzzentrum.de/uploads/sdm/SDM-Methodology_V2.0b.pdf [Accessed on 20 July 2022].

"Smart Contract". (2020). Smart Contract. [Available online: https://corporatefinanceinstitute.com/resources/knowledge/deals/smart-contracts/, Accessed on 20 July 2022].

"smashHit Demonstrator Videos". (2022). smashHit Demonstrator Videos [Available at https://smashhit.eu/demonstrator-videos/, Accessed on 05 June 2022].

"smashHit Whitepapers". (2022). smashHit Whitepapers [Available at https://smashhit.eu/whitepapers/].

"SPARQL, Endpoint URL". (2023). Sparql enpoint url [Available online: https://disorder-question-answer-interface.sti2.at/sparql, Accessed on 15 July 2023].

"SPARQLWrapper". (2008). SPARQL Endpoint Interface to Python. [Available online: https://sparqlwrapper.readthedocs.io/en/latest/, Accessed on 8 June 2022].

Summary and Future Work

The primary objective of this research is to improve data sharing infrastructures with contracts, licenses and building on and of semantic technology in domains such as smart cities, insurance, and healthcare. To achieve the objective of the research and to overcome the issues and challenges discussed in Chapter 1, the research problem has an RQ, which is further divided into five RSQs. The main RQ is answered by the answers of five RSQs. The summary of answers regarding RSQs is described as follows:

RSQ1 *How can semantic contracts be modeled and improved using an ontology as a data model?*

The RSQ1 is addressed by the development and enrichment of data sharing contracts using semantic technologies, resulting in a smashHitCore ontology (Kurteva et al., 2023). The smashHitCore ontology not only provides the semantic model of contracts but also the semantic model of consent and sensor data. More specifically, FIBO and MTCO are used for enriching and modeling contracts for data sharing. Further, the ontology is based on two real-world UC1 (The smashHit project, 2020b) and UC2 (The smashHit project, 2020c) in the smart cities and insurance domains. In addition, the smashHitCore ontology is evaluated against the aforementioned use cases by validating a set of competency questions regarding contracts and by standard ontology evaluation tools (see more details in Chapter 2).

RSQ2 *How can the CCV comply with GDPR and the CLM be performed to improve digital contracting services?*

The RSQ2 is addressed by implementing a scaleable and interoperable software application/tool for automated GDPR-compliant CCV (Tauqeer et al., 2022) that addresses

GDPR-specific challenges regarding data sharing and improves digital contracting services through contracts. The tool provides an extended version of the semantic model of contracts for data sharing based on UC1 (The smashHit project, 2020b) and UC2 (The smashHit project, 2020c) and is used in compliance verification along with GDPR. The legal requirements were derived from both use cases and translated into code. In addition, while the tool follows the principles of data protection by default, TOMs were implemented, which are a key requirement of GDPR. There were four CCV scenarios (Tauqeer et al., 2022) tested and evaluated against the use cases in the smart cities and insurance domains. Further, the tool supports scalable and interoperable processing and integration with other third-party software components and services. Moreover, the evaluation of the tool in terms of execution time and the TOMs were also performed.

RSQ3 *How can multi-legal-based GDPR-compliant data sharing be enabled by employing consent, contracts, and licenses with SHACL?*

The RSQ3 is addressed by extending our previous work, the CCV (Tauqeer et al., 2022), which enables data sharing to comply with GDPR through consent and contracts only. We added a new legal base "license-based contracts" that fully comply with GDPR. The license-based contract uses the standard licenses from the DALLIC framework and has its own additional contractual obligations. This combination gives more control over the PID to the contractors. Another useful feature is the use of SHACL validation, which is more semantically compliant, easily extendable, and supports integration (Corman et al., 2018). In addition, other functional requirements of the CCV tool, such as the implementation of digital signatures, and the performance evaluation of the tool in terms of execution time and testing the correctness of SHACL validation results are also major outcomes for this extended version of the CCV tool.

RSQ4 *How can the gap between users' questions and SPARQL queries over QAKG be narrowed by employing UI?*

The RSQ4 is addressed by extending/enriching the existing ontology CSTD and the creation of smell and taste disorder KGs to support and improve data sharing in the domain of healthcare (Tauqeer et al., 2023). The CSTD is enriched with real patients' data from the hospital "Smell and Taste Center, Hospital Gelderse Vallei, the Netherlands" and transformed into KG for smell and taste disorders. This KG is used for questioning-answering in natural language and with SPARQL queries. In addition, a UI helps the end users (who do not have experience in SPARQL queries) to make questions and get answers over the KG graphically and with ease. Information regarding the extended version of the ontology and the newly created smell and taste disorder KGs can be found in Chapter 5.

RSQ5 *How can the KV on personal assistant and knowledge bases be improved for checking the correctness, improvement, and quality of knowledge?*

Finally, the RSQ5 is addressed by implementing a graph validator application to perform KV on personal assistants and search engines (e.g., Google, Yandex) to ensure the quality of knowledge, which is measured by the degree to which statements are semantically corrected (Huaman et al., 2021). A confidence score is computed for each semantic triple and instance in KG. A weight is assigned to each weighted source (i.e., Google, Yandex). The determined score is based on discovering the same cases across different weighted knowledge sources and comparing their attributes/features (e.g., name, phone).

The RSQs' answers conclude that the proposed research and developed solution are successfully validated and enable GDPR-complaint data sharing by employing consent, contracts, and licenses through KGs. However, there are many limitations and we put them on a list for future work, as follows.

- The solution is successfully validated against smart cities and insurance domains, however, other domains like healthcare, sports, and education can also be strong candidates for data sharing. As an example, our work (Tauqeer et al., 2023) can be extended with the proposed solution for data sharing to comply with GDPR.
- This solution uses only a generic template for smart cities and insurance domains, however, there is a need for other templates in domains like education, healthcare, and sports.
- The developed solution improves the CLM stages, particularly, the auditing/controlling stage and the signing stage. The improvement in the negotiation stage by adding the integration of social communication channels such as Slack [24] and collaboration tools like Google Docs [25] are included in future work.
- In addition, to improve the execution time of CLM, future work includes the enhancement and improvement in the creation stage of the CLM by providing the mostly used contractual terms and clauses from the contract repository (i.e., KGs).
- Lastly, the implementation of a UI, which is used to manage contracts, licenses, and compliance verification checks is also included in future work.

[24]https://slack.com/
[25]https://www.google.com/docs/about/

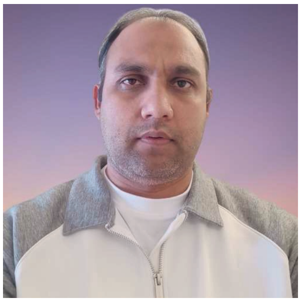
Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor and promotor Prof. Dr. Anna Fensel, for her invaluable supervision, patience, input, advice, feedback, guidance, and continuous support throughout my PhD research project. I greatly thank Anna, for her immense knowledge and valuable experience that have encouraged me all the time in my academic research, and also for giving me the opportunity to work in her team. I would also thanks to Univ.-Prof. Dr. Dieter Fensel and the whole STI Innsbruck team at the University of Innsbruck that supported me in my PhD. Further, I would like to express my gratitude to the whole WASS administration team at the Wageningen University & Research, the Netherlands. More specifically, I greatly thank Heleen Danen-Louwerse and her team for managing administrative tasks at WASS.

Furthermore, this research would not have been possible without the support from the project smashHit within the H2020-EU "INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT) with grant agreement number 871477", which financed my research.

I would also like to appreciate all the discussions I had with Tek Raj Chhetri, Anelia Kurteva, Sareh Aghaei, and Elwin Huaman. Finally, I would like to express my gratitude to my parents, my wife, and my children. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

About the Author



Amar Tauqeer was born on the 15th of October 1984 in Wazirabad, Gujranwala, Pakistan. He obtained his bachelor's degree "Bachelor of Computer Science" from the AIOU (Allama Iqbal Open University), Islamabad, Pakistan (2003-2005), where software development was his specialization. During his undergraduate studies, Amar developed a passion for software development and decided to take the next steps in forging his career in this area. For this purpose, he joined a leading software development company PMS(Pvt), Ltd., Gujranwala, Pakistan, and worked there from 2006 to 2009. After getting industrial experience, he earned his master's degree "Master of Science in Information Technology" at the UOG (University of Gujrat), Gujrat, Pakistan. To explore knowledge and pursue higher studies abroad, he did another master's degree "Master in Information Systems" at WU (Vienna University of Economics and Business), Vienna, Austria. During his master's, he developed an information system "Visualizing-Geo-located-Open-Data" to show location information (e.g., address, phone numbers, email, and web links) about Vienna's districts' data like parking spots, rail stations, hospitals, etc. After graduating in 2017, he worked as a freelancer and working student for more than two years, where he worked on different software projects such as "The Citizen International Plastic Industries (Pvt) Ltd., Gujranwala, Pakistan", and "The University of Innsbruck, Innsbruck, Austria". He started with his PhD project studies and developments in 2020 and enrolled as a PhD student at Wageningen University & Research. During his PhD work, he investigated on the topic of improving GDPR-compliant data sharing through consent, contracts, and licenses using semantic technologies. Further, he developed an automated GDPR-compliant tool for improving data sharing in fields like smart cities and insurance. The results of this PhD research are presented in this thesis.

Publications

Peer-reviewed journal publications

1. **Tauqeer A.**, Kurteva A., Chhetri T.R., Ahmeti A., Fensel A. Automated GDPR Contract Compliance Verification Using Knowledge Graphs. Information. 2022; 13(10):447. https://doi.org/10.3390/info13100447

2. **Tauqeer A.**, Hammid I., Aghaei S., Parvin P., Postma E.M., and Fensel A. Smell and taste disorders knowledge graph: Answering questions using health data. Expert Systems with Applications, page 121049, 2023. https://doi.org/10.1016/j.eswa.2023.121049

3. Kurteva A., Chhetri T.R., **Tauqeer A.**, Hilscher R., Fensel A., Nagorny K., Correia A., Zilverberg A., Schestakov S., Funke T., Demidova E. The smashHitCore Ontology for GDPR-Compliant Sensor Data Sharing in Smart Cities. Sensors. 2023; 23(13):6188. https://doi.org/10.3390/s23136188

4. **Tauqeer A.**, Fensel A. GDPR Data Sharing Contract Management and Compliance Verification Tool. Software Impacts, 100653. https://doi.org/10.1016/j.simpa.2024.100653

5. **Tauqeer A.**, Chhetri T.R., David R., Ahmeti. A., Fensel A. An Integrated Approach to GDPR-compliant Data Sharing Employing Consent, Contracts, and Licenses (Submitted to Data & Knowledge Engineering journal, under review.)

Peer-reviewed conference publications

6. Huaman, E., **Tauqeer A.**, Fensel A. (2021). Towards Knowledge Graphs Validation Through Weighted Knowledge Sources. In: Villazón-Terrazas, B., Ortiz-Rodríguez, F., Tiwari, S., Goyal, A., Jabbar, M. (eds) Knowledge Graphs and Semantic Web. KGSWC 2021. Communications in Computer and Information Science, vol 1459. Springer, Cham. https://doi.org/10.1007/978-3-030-91305-2_4

7. David R., Ahmeti A., Fensel A., **Tauqeer A.** SHACL-based Application of Integrity Constraints and Repairs for Contract Compliance Verification (Submitted, under review.)

Other peer reviewed publications

8. Kurteva A., **Tauqeer A.**, Fensel A. (2022). Data Sharing in smashHit: Making Consent and Contracts Interpretable with Knowledge Graphs, Trusted Secure Data Sharing Space (TRUSTS), Workshop on Data Spaces and Semantic Interoperability, 2022, Available at https://www.trusts-data.eu/wp-content/uploads/2022/06/06-Data-sharing-in-smashHit-Making-consent-and-contracts-interpretable-with-knowledge-graphs.pdf

WASS Training and Supervision Certificate

Amar Tauqeer
Wageningen School of Social Sciences (WASS)
Completed Training and Supervision Plan

Name of the learning activity	Department/Institute	Year	ECTS*
A) Project related competences			
A1 Managing a research project			
WASS introduction course I and II	WASS/ WUR	2021	1
Transfer Learning: A paradigm for machine-assisted knowledge transfer	CCNC 2022 Workshop II-WOT	2021-22	1
Interoperability and Scalability Trade-offs in Open IoT Platforms	CCNC 2022 Workshop II-WOT	2021-22	1
HOLY: An Ontology covering the Hydrogen Market	ESWC Conference	2023	1

Name of the learning activity	**Department/Institute**	**Year**	**ECTS***
Conference presentation, oral or poster presentation at KGSWC 2021, further research work presented a number of further times such as at WUR, EU project smashHit review meetings, and final project review.	KGSWC 2021, Wageningen University & Research, smashHit project	2022-23	2
A2. Integrating research in the corresponding discipline			
703652 VO Semantic Web	Computer Science/ University of Innsbruck	2020-21	6
703340 SE Research Seminar: Semantic Technology	Computer Science/ University of Innsbruck	2021	5
703653 PS Semantic Web	Computer Science/ University of Innsbruck	2020-21	4
B) General research related competences			
B1. Placing research in a broader scientific (social sciences and WUR) context			
800922 VU Text and Web Mining mit RapidMiner	Office of the Vice Rector for Teaching and Students/ University of Innsbruck	2022	2.5
800960 VU Data management and analysis for environmental sciences	Office of the Vice Rector for Teaching and Students/ University of Innsbruck	2022-23	4
B2. Placing research in a societal context			
Writing a blog (for smashHit project: https://smashhit.eu)	Computer Science/ University of Innsbruck	2022	1
C) Career related competences/personal development			
C1. Employing transferable skills in different domains/careers			
800909 SE Successful Meetings	Office of the Vice Rector for Teaching and Students/ University of Innsbruck	2021-22	2.5
Supervision – Amar Tauqeer acted as a local host supervisor of a WUR MSc student Ismaheel Hammid, who completed his Master study internship at the University of Innsbruck in spring-summer 2022.	University of Innsbruck and Wageningen University & Research	2022	4
Total			**35**

*One credit according to ECTS is on average equivalent to 28 hours of study load

Colophon

The research described in this thesis was financially supported by the Horizon 2020 project smashHit (grant number 871477) and partially supported by the Eureka, Eurostars program project WordLiftNG (grant agreement number 877857 with the Austrian Research Promotion Agency (FFG)) (work of Sareh Aghaei and Anna Fensel), the Netidee scholarship 2022 by the Internet Foundation Austria (work of Sareh Aghaei), as well as by Wageningen University & Research. Further, financial support from Wageningen University for printing this thesis is gratefully acknowledged.

Amar Tauqeer designed the cover pages and the template used for the cover pages was downloaded from Canva's book cover template (Blue Modern Business Proposal Cover).

Publisher: ProefschriftMaken

WAGENINGEN
UNIVERSITY & RESEARCH

Wageningen School
of Social Sciences