Jheronimus Academy of Data Science & Info Support
Data Science in Business & Entrepreneurship

Toward Automated Data Contract Enforcement: An LLM-based Knowledge Graph Construction Approach

Master's Thesis

T.D. Hoven

Supervisors:
Dr. Indika Weerasingha Dewage (JADS)
Dr. Geert Monsieur (JADS)
Yven Lommen (Info Support)

's-Hertogenbosch, July 2026

Abstract

Data contracts promise reliable, governed data exchange in decentralized architectures, yet remain difficult to enforce in practice. Although a standard specification is emerging, its low maturity and limited adoption mean data contracts are still authored in various formats. Technical engineers must translate these into enforceable code through tools or manual labor, creating a bottleneck in a growing data landscape.

This research investigates how Large Language Models (LLMs) can help to enforce data contracts. Following a Design Science Research paradigm and grounded in 11 semi-structured interviews with industry experts, it proposes the LLM-Assisted Contract Extraction (LACE) Framework to integrate the data products. Rather than enforcing contracts directly, the framework uses an LLM as a translation layer that converts them into a deterministic, machine-readable intermediary model, expressed as a Resource Description Framework (RDF) knowledge graph against a purpose-built Data Contract Ontology. SHACL validation safeguards its structure, and a provenance lineage model preserves a human-in-the-loop.

The artifact was assessed through an ontology evaluation, an intrinsic evaluation, and an extrinsic evaluation with six practitioners. The results show that structural conformance, rather than content retrieval, is the dominant failure mode, and that example-based prompting is needed for usable, reproducible output. Practitioners valued the framework most for its standardization, locating the main adoption barriers in organizational rather than technical concerns. LLMs, therefore, help to enforce data contracts not by enforcing them directly, but by transforming a manual, error-prone translation step into a validated, repeatable foundation for downstream governance automation.

Preface

This thesis marks the end of my master's in Data Science in Business and Entrepreneurship at the Jheronimus Academy of Data Science. I am glad to conclude my studies with a topic that sits at the intersection of an industry practice and an emerging technology.

Because this thesis spans many parts, I have learned a lot along the way: from data contracts and decentralized data architectures such as the data mesh to ontology engineering and knowledge graphs. Speaking to so many industry experts was especially valuable, as it showed me how organizations actually work with their data and the challenges they face.

First and foremost, I would like to express my gratitude to my supervisor, Dr. Indika Weerasingha Dewage, for his close guidance throughout my thesis journey. His involvement significantly improved my thesis through his feedback, recommendations, and the interesting discussions we had about the topic. I am particularly grateful for his close guidance and quick responsiveness to my questions. I would also like to thank my second supervisor, Dr. Geert Monsieur, for his feedback on my final draft.

I am also very grateful to Info Support for giving me the opportunity to carry out this research with their resources and the ability to get training in specific domains. Special thanks to my company supervisor, Yven Lommen, for his openness in sharing his expertise, his practical feedback on the research, and his willingness to challenge my assumptions and approaches. I genuinely enjoyed my time here.

Finally, I would like to thank all the research participants who took part in the interviews and the evaluation of the artifact. Without your help, the artifact would never be where it is now, and your input and perspectives were invaluable. The LACE Framework rests on your practical experience as its foundation.

I am grateful for all the help I received throughout this process, and proud of the final result.

T.D. Hoven
's-Hertogenbosch, July 2026

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Research Context

Since many organizations began using, producing, and consuming vast amounts of data, it became essential to rely on the data stored. This data is often used to develop strategies, tactics, and tools that create a competitive advantage and improve operational productivity. Ghasemaghaei and Calic have shown that high variety, volume, and velocity significantly impact innovation performance [39]. To handle such volumes of varied data and create value from it, Dehghani proposed a new decentralized architecture called the data mesh [25]. The data mesh is a domain-driven architecture in which a domain within an organization is responsible for creating data products. Consumers, other data products, domains, or end-user applications consume the data produced by the data product. This approach allows organizations to standardize, combine, and reuse their data assets, making them more agile and effective in implementing data-driven decisions [14].

Different challenges arise when working with data meshes or data spaces, including reliability, quality, and data governance [60]. In 2020, Gartner reported that poor data quality costs companies an average of $12.9 million per year [37]. Data quality is defined by how well the data fulfills the specific needs and standards set by the organization. Based on the ISO/IEC 25012 data quality model[1], data quality is typically evaluated using characteristics such as Accuracy, Completeness, Consistency, and Accessibility. Closely related to this is data reliability, which serves as a prerequisite for ensuring that the underlying data is trustworthy for decision-making. Data reliability can be evaluated through internal reliability (data conforms to global characteristics), relative reliability (how well the data aligns with organizational policies), and absolute reliability (evaluates the degree to which the data reflects real-world facts) [5].

To address these quality and reliability problems, organizations must improve their data governance, the practice that determines who holds decision rights over data assets and how authority and control over data management are exercised [64, 56]. In the data mesh, some of these problems are solved through domain-driven design. In other data architectures, challenges such as the lack of expectations (consumers do not know what the data means or how to use it), the lack of reliability (the data often changes or breaks without warning), and the lack of autonomy (teams depend on a central group to access or manage data) persist [58]. To address these issues, Dehghani also introduced data contracts [25], agreed-upon interfaces that set expectations for data producers and consumers. It defines how the data product should be governed and enforces data quality, ensuring it explicitly meets business requirements [58].

To ensure that business requirements are met, data producers and consumers formalize data contracts as human- and machine-readable agreements. These contracts define Service Level Agreements (SLAs) and operational parameters, including data product schemas, data quality standards, regulatory policies, and ownership rights [103]. By defining these dimensions, data contracts enable automated validation, such as schema compliance and timeliness thresholds. This ensures that those constraints are enforced before data is consumed or

[1] https://www.iso.org/standard/35736.html

processed.

Through this formalization, data contracts provide a practical mechanism for addressing challenges in decentralized architectures. They facilitate standardization across domains and improve data discoverability and trust by exposing the data product's metadata, and ensure compliance with security and governance protocols through the contract [73]. By establishing agreed-upon expectations, these contracts allow domains to manage their data autonomously and accelerate data-driven innovation [73].

1.2 Problem Statement

Despite this promise, a major shortcoming of data contracts is that they come from industry rather than academic research. Given this origin, there is little research and literature on the subject; therefore, many research challenges remain. There is an ongoing effort to develop a standard data contract specification [13], but due to its low maturity, limited scientific research, and limited adoption in organizations, it has not yet been widely implemented [103]. The problem arises because the teams behind the implementation determine how a data contract is structured (e.g., JSON or YAML) [58]. Many organizations do not enforce their data contracts, and the few that do use a variety of technologies to enforce them. This lack of standards creates a significant bottleneck: various organizational roles may draft the contracts, while technical engineers are left responsible for translating these agreements into executable enforcement mechanisms and therefore rely on manual implementations through coding and tool configurations, which severely limit scalability and consistency across the current data landscape.

1.3 Research Objective

To overcome this operational barrier, we propose an LLM-Assisted Contract Extraction (LACE) framework designed to translate data contract specifications into a structured, intermediary unified model. This research focuses on how LLMs can be used to create a deterministic foundation required for automating data contract enforcement. To establish a deterministic foundation, this framework will generate a Knowledge Graph (KG) that provides downstream applications with the context needed to produce accurate enforcement code or to answer questions about the available data products and contracts. Additionally, using this KG structure helps address the previously mentioned challenges by providing an accessible, discoverable data layer. Organizations can use this layer as a centralized registry for their data products, integrating metadata and governance rules. This allows for data contracts to be machine-readable, which ultimately allows their enforcement to be automated, consistent, and predictable.

1.4 Research Questions

To explore the viability and design of this proposed framework, this thesis addresses the following research questions:

Main Research Question: How can LLMs help to enforce data contracts?

Sub-Research Questions (SRQ):

SRQ1: What are the primary technical and organizational barriers for enforcing governance within decentralized data architectures?

By researching the obstacles practitioners face, this question grounds the artifact in the real bottlenecks of governance enforcement, ensuring the design targets the barriers that block adoption rather than assumed ones.

SRQ2: How are data contracts currently implemented and enforced in the industry?
Documenting how data contracts are written and enforced in practice exposes the problems that the framework must resolve, grounding its design in current industry reality.

SRQ3: What architectural principles and mechanisms are required to ensure the trustworthiness and reliability of LLM-extracted data contract information?
Because LLMs are non-deterministic, this question identifies the principles and safeguards needed before LLM-extracted information can be trusted by organizations.

SRQ4: What components and interactions are required to automate the governance surrounding data contracts & products?
By determining which components and interactions automate governance, this question moves the isolated extraction to a realistic framework, clarifying where the LLM sits relative to the enforcement decision and what makes the framework usable.

SRQ5: What is the impact of different prompt engineering techniques on LLM-extracted data contract information?
By examining how prompting techniques shape the output, this question reveals which prompt techniques yield accurate, valid, and reproducible extractions. Reliable output is a core determinant of whether the framework can be populated at a usable quality level.

1.5 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 establishes the theoretical foundation through a literature review. Chapter 3 outlines the overarching research design and details the methodology used in this research. In Chapter 4 the problem is explored and identified through practitioner interviews, creating a problem statement and proposing design requirements for the artifact. Chapter 5 details the design and development of this artifact, focusing on ontology engineering, the extraction pipeline, and the integration into a decentralized data architecture. Then, in Chapter 6, we discuss the evaluation methodology and environment for assessing the artifact, with the results presented in Chapter 7. Chapter 8 discusses the implications and limitations of the findings, while Chapter 9 addresses the threats to validity and reflects on how these affect the interpretation and generalizability of the findings. Finally, Chapter 10 presents the overall conclusions and provides directions for future research.

Chapter 2

Literature Review

This chapter establishes the theoretical foundation for the proposed LACE Framework in three parts. The background (Section 2.1) introduces the foundational concepts to understand the domain: the data mesh as the decentralized data architecture paradigm, the principles of federated computational data governance, and the definition and importance of data contracts are examined through the existing literature, and KGs and ontologies are discussed as the framework's structured representation of machine-readable data contracts. Section 2.2 discusses existing work that pursues similar goals, namely, automated policy enforcement in decentralized governance and the construction of KGs from heterogeneous text. Finally, the research gaps (Section 2.3) identify the limitations and shortcomings in the literature, establishing the basis for the methodology and research questions of this thesis.

2.1 Background

2.1.1 Data Mesh

The data mesh is a domain-driven, decentralized data architecture that aims to minimize or avoid operational bottlenecks associated with centralized and monolithic data architectures in enterprises [25, 42]. Earlier analytical paradigms, from the data warehouse to the data lake, concentrate the responsibility for ingesting, transforming, and serving data within a single specialized team [67]. As the number of data sources and data consumers grows, this central team becomes a bottleneck for increasing value from domain data. Another problem is that this team operates at a distance from the domains that produce the data and lacks the context required to interpret and maintain it correctly [42]. Dehghani frames the data mesh as a paradigm shift that decomposes the monolithic analytical data space into data domains aligned with business domains [24]. With this move, the responsibility for high-quality data and valuable insights shifts away from a central team and towards the team that knows the data best [42, 67]. This change is both structural and organizational, as data teams are arranged around specific domains, thereby becoming decentralized [67].

To ensure this decentralization does not lead to fragmentation, Dehghani defines the data mesh through four interdependent principles: domain ownership, data as a product, a self-serve data platform, and federated computational governance [24]. Figure 2.1 depicts how these principles combine in the logical architecture of the data mesh. Under domain ownership, analytical data is assigned to the business domains closest to it, so that the teams that best understand the data are accountable for storing and serving it [24]. The principle of data as a product applies product thinking to analytical data, as each domain should treat its data as a product whose quality must be assured and whose consumers are regarded as customers whose expectations must be met, and whose experience must be optimized [25].

The remaining two principles provide the shared foundation that keeps autonomous domains linked. The self-serve data platform provides the tooling and interfaces that enable domain teams to build, deploy, and maintain their data products without requiring engineering expertise. Abstracting infrastructure complexity allows domains to be more autonomous and manage their own work [24]. The final principle, federated computational

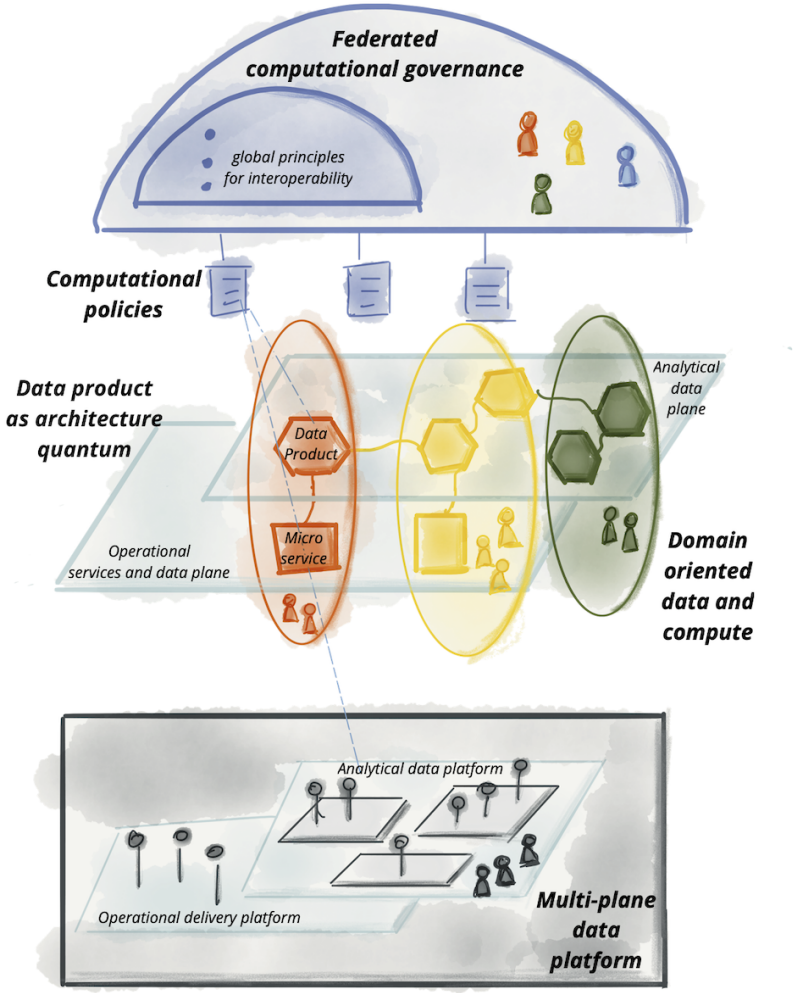

Figure 2.1: Logical architecture of the data mesh from Dehghani [24].

governance, complements this self-serve data platform by establishing a set of global rules that are applied automatically across all data products and their interactions. This allows for interoperability between separate data domains, and regulations and compliance rules can be preserved across the data mesh [24]. Because this fourth principle is the direct foundation of the automated enforcement addressed in the present work, it is examined in greater detail in the following section.

Taken together, these principles describe a network of interoperable data products exchanged across autonomous domains under a set of common governance standards.

2.1.2 Federated Computational Data Governance

Federated computational governance, the fourth principle of the data mesh, determines how global policies are defined and automatically enforced across otherwise autonomous domains. Understanding this principle first requires situating it within the evolution of data governance from which it inherits its core concerns.

The organizational paradigm has evolved to treat data not as a byproduct of operations but as a critical asset [64]. This perspective has reinforced the recognition of data as a strategic asset, driving increased interest in data management, governance, and quality assurance [56]. Numerous data governance frameworks have emerged due to the interest in governing data assets. Khatri & Brown adapted the IT governance principles of Weill & Ross to establish one of the earliest frameworks for data governance, organizing it into five decision domains (Figure 2.2) [64]. In the last decade, additional data governance frameworks have emerged in the literature. Regardless of scope, a recurring purpose across these frameworks is to balance mitigating the risk of unauthorized data use with promoting data use to advance various interests and goals [69]. While these frameworks provide a theoretical basis for data governance, operationalizing them remains a challenge due to the diversity of technologies, domains, and organizational roles involved [72].

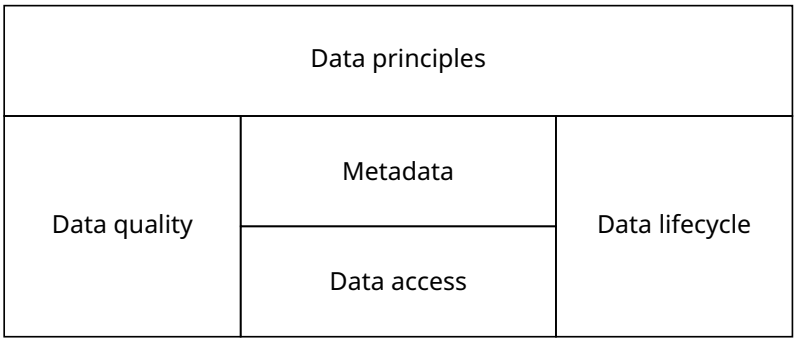

Figure 2.2: The five decision domains for data governance, adapted from Khatri & Brown [64].

In decentralized data architectures, such as the data mesh, traditional governance approaches often fail to address the complexities of ensuring data consistency and interoperability across distributed domains [85]. To tackle this, Dehghani's fourth principle of data mesh explicitly calls for "federated computational data governance," in which data governance is enabled by automation through the data mesh platform [25]. Kanagarla emphasizes that shifting to a decentralized data mesh introduces significant complexities regarding cross-domain data consistency, federated governance implementation, and cultural resistance to decentralized data ownership [60]. To successfully navigate these challenges, organizations must invest heavily in scalable, self-serve technical infrastructures and provide training to increase the autonomy of domain teams [60]. This is often the biggest challenge professionals face, as they are concerned about automated execution, especially in security, regulatory, and privacy-related areas [15].

2.1.3 Data Contracts

In the context of decentralized data architectures such as the data mesh, data contracts serve as the critical mechanism for ensuring reliable, secure, and high-quality data exchange across domains. While academic research is sparse, the existing literature provides sufficient perspective to develop a useful definition of data contracts.

At a fundamental level, data contracts are defined as formal agreements between data producers and consumers that outline data ownership, terms of data offering and use, and guarantees [103, 73]. Within a data mesh, they serve as formal, human- and machine-readable interfaces between dataset producers and an arbitrary number of data consumers [73]. By explicitly describing the characteristics of data delivered by a data product, the constraints applied, and the terms of use for potential consumers, data contracts empower business domains to manage their own data through bilateral agreements [103].

To fulfill this role, a data contract typically comprises three core components [73]:

- Information about the contract (e.g., identifiers, domain) and dataset (e.g., columns, quality assurance).
- All necessary information to access (e.g., protocols, URL), understand (e.g., SLAs), and use (e.g., terms of use, governance rules) the data.
- Information about the data producers and consumers (e.g., owner, names).

Expanding upon these high-level components, a systematic review of data mesh literature reveals a granular set of specific elements necessary for a fully operational data contract [103]:

- **Location and Access Requirements:** The contract must define the data product's location and specify how to access it through its endpoints, including the exact queries consumers are permitted to send.
- **Schema and Semantics:** To ensure accurate and reliable usage, the contract explicitly defines the data model (structure and format) and the semantics, which helps consumers consistently understand the delivered data.
- **Data Quality Standards:** Explicit expectations for quality dimensions, such as accuracy and completeness, are established to prevent downstream consumers from experiencing quality issues and to increase overall data confidence.
- **Policies and Regulatory Compliance:** Contracts specify policies for data encoding, anonymization, and retention. This ensures that operations on sensitive data comply with regulatory requirements, such as the GDPR.
- **Ownership and Accountability:** By clearly assigning ownership across the data value chain, contracts ensure the correct personnel can be contacted in the event of data errors.
- **Service-Level Agreements (SLAs):** Data contracts outline the terms of service, embedding guarantees regarding data quality, availability, and standard software quality attributes like response time.
- **Versioning and Auditability:** To protect consumers from rapid, unexpected changes, data contracts must be versioned. Furthermore, modifications should be recorded in an audit trail to support transparency and accountability.
- **Pricing and Use Cases:** Where applicable, contracts can outline the potential use cases and terms of use, as well as define the billing process for data product consumers.

From an architecture modeling perspective, the data contract is positioned as an essential companion to the data product itself [97]. Within the domain-driven data creation process, operational data is transformed into a data product that must be accompanied by a data contract. This contract ensures that the created data product complies with federated governance policies before it is officially published in the data product catalog for consumption. Consequently, data contracts are instrumental in ensuring high data quality, facilitating standardization across different domains, and defining SLA guarantees regarding data availability [103, 73].

Wasser et al. identify two types of data contracts: data product APIs and data-sharing and usage agreements [103]. When functioning as APIs, data contracts are integrated into the input and output ports of data products. They serve as machine-readable specifications that outline data consumption methods and technical constraints that apply. The other variation, data-sharing and usage agreements, defines the relationship between data providers and consumers, detailing the specific purpose and terms of data use. These mutual agreements

are established once a consumer's access to a data product is approved. To increase more "product thinking", either party can end the agreement if certain conditions, such as poor data quality or insufficient business value, are met. These agreements serve as the foundation for managing data access permissions [103].

Furthermore, while foundational data mesh literature emphasizes technical and architectural interoperability, the concept of data contracts can also be viewed through the lens of regulatory compliance and ethics. Although predating the widespread adoption of the data mesh paradigm, Hadziselimovic et al. (2017) provide a different perspective by proposing a machine-readable data protection rights contract designed to support data protection and data ethics in the sharing of scientific data [46]. In this regulatory context, a data contract functions as a governance instrument ensuring that shared personal data is only used for the purposes agreed upon with the data subject, requiring cooperation in respecting data subject rights even after the data is shared, and enabling data protection compliance checks by the organization that shared the data [46]. This perspective introduces the need for formal languages, such as the Data Protection Rights Language (DPRL), to encode these obligations, highlighting that data contracts within a data mesh must not only govern technical data exchange but also rigorously enforce data privacy and legal compliance across distributed domains [46].

2.1.4 Knowledge Graphs & Ontology

In decentralized data architectures and automated governance, KGs have emerged as a mechanism for representing structured information, allowing machines to effectively process, query, and reason about complex data ecosystems [62, 110]. By organizing entities, concepts, and policies as nodes connected by semantic relationship edges, KGs store facts explicitly as *subject-predicate-object* triples.

Formally, a KG can be defined as a directed graph (G), where $G = (E, R)$ [92]. Here, E denotes the set of entities or nodes that serve as subjects and objects in the KG, where $E = \{e_1, \ldots, e_n\}$. R refers to the set of relationships or edges, $R = \{r_1, \ldots, r_m\}$. Together, the entities and relations are expressed as triples (*subject-predicate-object*), with the total graph represented as:

$$G \subseteq E \times R \times E$$

While this is the formal definition, a single unit of knowledge within this directed graph is denoted as the triple $f = (s, p, o)$, where:

- $s \in E$ represents the *subject* (the source entity)
- $p \in R$ represents the *predicate* (the directed semantic relationship)
- $o \in E$ represents the *object* (the target entity or value)

The triple (s, p, o) structure is often referred to as a *statement* or *knowledge triple* [94]. An example of a triple is [`CustomerData, ownedBy, SalesDomain`], indicating that the data product `CustomerData` is owned by the `SalesDomain`. By decomposing complex information into these statements, the KG provides a machine-readable framework for data integration and extraction [94]. Figure 2.3 illustrates how individual triples connect through shared entities to form a larger graph.

There are two main types of KGs: Labeled Property Graphs (LPGs) and Resource Description Framework (RDF) graphs. The first comes mainly from engineering communities, while the RDF graph technology is developed by the W3C to exchange data on the web [38]. Structurally, an RDF graph is a directed edge-labeled graph where a piece of information can only be represented as either a node or an edge. This structure makes the RDF model highly suitable for semantic web applications, knowledge representations, and inference. In contrast, the LPG model is a directed graph in which both vertices and edges can carry an

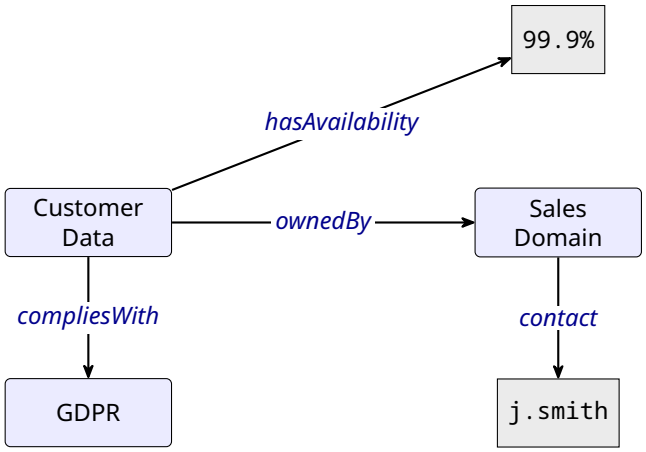

Figure 2.3: A knowledge graph built from *subject–predicate–object* triples.

arbitrary number of properties, enabling it to capture more information with fewer nodes and edges than the RDF model [95]. Figure 2.4 shows the contrasts between the two models by representing the same fact in each.

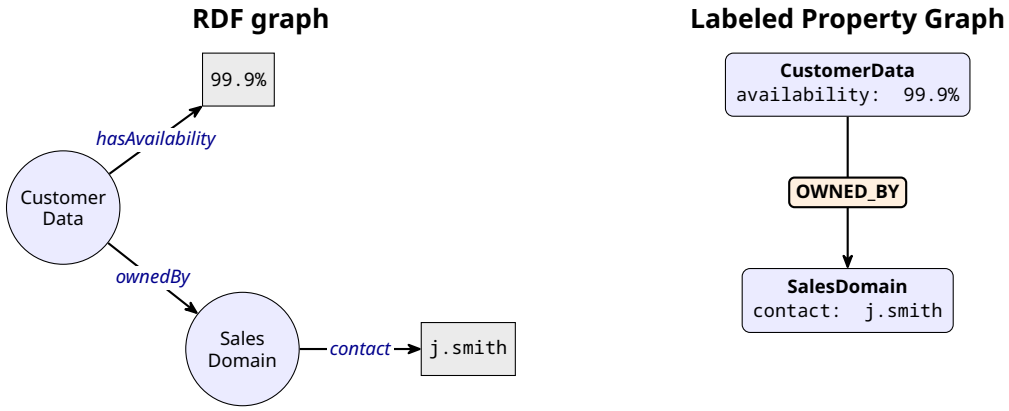

Figure 2.4: The same facts modelled as an RDF graph and as a Labeled Property Graph.

To enforce domain structure and explicit semantics in a KG, these graph technologies often integrate an ontology, a formal structure that organizes a system's relevant entities into a hierarchy of concepts and defines relationships among them [43]. RDF natively supports ontology modeling languages, allowing the graph to store formalized domain knowledge. With this knowledge, RDF can perform automated inference (generating new knowledge from existing data), which makes it highly usable in multi-domain and multi-stakeholder environments [27]. Native LPG architectures do not use schemas or ontology languages. This lack of a rigid schema leads to greater flexibility in LPGs, reduced programmatic complexity, and faster traversal performance [27].

While generic, broad-domain KGs have seen massive success, there is a growing recognition that "one-size-fits-all" ontologies are inadequate for specialized fields. Consequently, domain-specific KGs require specialized techniques for construction, entity resolution, and inference to capture the semantic nuances of domain-specific languages [62].

Traditionally, automated KG construction is framed as a three-step procedure: knowledge acquisition, knowledge refinement, and knowledge evolution [110]. This process often utilizes Natural Language Processing (NLP) techniques, such as coreference resolution, named entity linking, and relationship extraction, to generate actionable, machine-readable triples from

unstructured datasets [94]. The scope of these representations is also expanding beyond pure text, as Zhu et al. systematically reviewed Multi-Modal Knowledge Graphs, which use symbolic knowledge to non-symbolic experiences to achieve a more comprehensive understanding of real-world contexts [111].

2.1.5 Large Language Models

The recent progress in automated KG construction is largely driven by Large Language Models (LLMs), a class of neural networks designed to process and generate human language. LLMs are built on the Transformer architecture (Figure 2.5), which Vaswani et al. introduced as a model that moves away from step-by-step sequential processing of earlier architectures and introduces a mechanism called self-attention [100]. Before text can be processed, it is first divided into tokens, the atomic unit an LLM operates on. The splitting of text is not done through whole words, but by tokenizing parts of each word, where frequent words are represented as a single token while rarer words are often broken into smaller fragments [100]. This keeps the vocabulary compact while ensuring that any input, including previously unseen words, can still be expressed as a sequence of known tokens. Each token is then mapped to a numerical vector (embedding), that the model can process [100]. By allowing every token to attend directly to every other token in a sequence, self-attention captures long-range dependencies while remaining highly parallelizable, making it feasible to train models at scale, which was previously impractical [100]. Building on this architecture, LLMs are trained on massive text corpora using self-supervised objectives such as autoregressive next-token prediction, in which the model repeatedly predicts the next token in the sequence that precedes it. Through this objective, the model learns general linguistic and factual patterns without manually labelled data [17]

The capability of these models is closely tied to their scale. Kaplan et al. demonstrate that model performance scales as a power law of model size, dataset size, and training compute [61]. In practice, this means that increasing the scale of a model yields better performance while making training more efficient. These findings motivated the development of increasingly large models, which, as they grow, exhibit new capabilities like reasoning and improved question answering [61]. Brown et al. illustrate this with GPT-3, an autoregressive model with 175 billion parameters that performs a wide range of tasks without fine-tuning, enabling a single model to be applied to specialized tasks [17].

Because LLMs are steered through natural language rather than retraining, the way a task is specified directly affects the quality of the output. Instead of fine-tuning, performance can be improved by providing clear prompts that define the task or provide real examples, a practice known as in-context learning, in which the model uses a handful of examples specified purely through text [17]. The design of these prompts, known as prompt engineering, has therefore become central to correct and reliable output. Methods like chain-of-thought, where the prompt includes intermediate reasoning steps as examples, or one-shot or few-shot, where the prompt includes one or more examples, substantially improve performance on arithmetic, logic, and reasoning tasks, and the effect emerges primarily in larger LLMs [104].

While LLMs are primarily trained on unstructured text, these capabilities also enable them to generate structured output. Relying solely on prompting techniques for this task is unreliable due to the non-determinism of the output. A common approach to obtaining more consistent results is to fine-tune a model on task-specific examples. As fully retraining a model with billions of parameters is computationally expensive, parameter-efficient methods such as Low-Rank Adaptation (LoRA) are frequently employed; these keep the original model weights frozen and instead train a small set of low-rank matrices, which drastically reduces the number of parameters that must be updated [51]. An alternative direction augments the generation process with additional, smaller models that refine or verify the output [47].

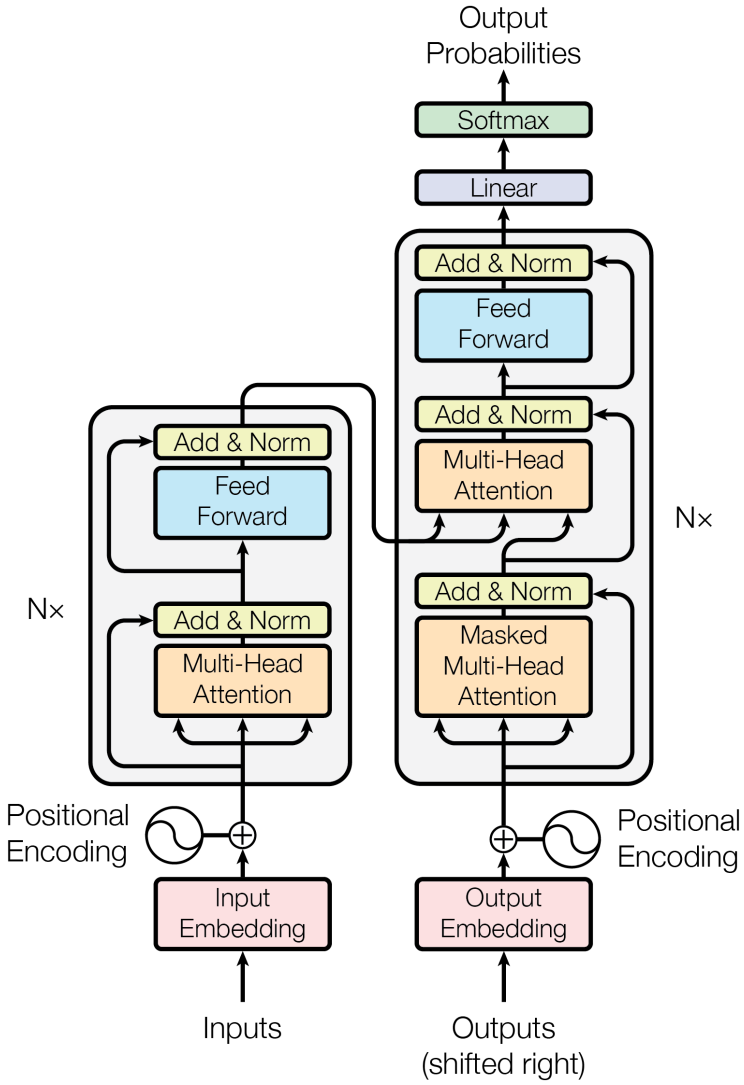

Figure 2.5: The Transformer model architecture from Vaswani et al. [100]

2.2 Related Works

Building on these foundations, this section presents existing research addressing challenges relevant to this study. These efforts fall mostly into two categories: automating policy enforcement and constructing KGs from heterogeneous or unstructured text.

2.2.1 Automated Policy Enforcement

The first category of work targets the automated enforcement of governance policies across decentralized domains. Wider et al. propose a platform-centric approach to data access control designed to reduce the tension between global standards and local decision-making [106]. A central authority defines global policies, while domain teams are responsible for tagging their data accordingly. They propose using the data platform to automate policy enforcement via transitive metadata tagging, in which data products inherit tags from their upstream sources. Regarding technical enforcement, Wider et al. identify two primary categories of policy automation. The first involves defining data sensitivity through metadata tags that propagate through the data lineage. This approach allows for consistent enforcement while granting local domains the autonomy to change the sensitivity levels. However, to combat that, everyone can override the sensitivity; any reduction in the sensitivity level should be reviewed and approved. The authors further discuss access control, outlining three distinct mechanisms: a Gateway model (centralized proxy), a Token model (decoupled authorization), and an Encryption model (key-based access). The authors conclude that effective decentralized governance requires combining these three approaches under an abstraction layer to minimize users' cognitive load [106].

Taking automated enforcement a step further, Garg proposes unifying data mesh principles with decentralized autonomous agents through a policy-as-code framework [36]. By defining business rules, regulatory requirements, and ethical constraints in declarative languages, organizations can consistently apply policies, audit through versioning, enable policy updates, and explain policies in a human-readable way.

Wider et al. also proposed a way to automate data access requests by incorporating Generative AI to analyze requests and evaluate them directly against plain-text global policies. By utilizing LLMs, the system assesses the context of the access request alongside relevant legal and privacy documents. This system can flag potential violations without requiring policies to be manually translated into formal rule-based contracts. Instead of fully replacing data stewards, this AI approach provides targeted warnings and remediation suggestions, ensuring that a human is still accountable for data access approvals [105].

2.2.2 Knowledge Graph Construction

A second category of work applies KGs to specific domains and increasingly relies on LLMs to automate the construction. The utility of domain-specific KGs has expanded rapidly across disciplines, highlighting their capacity to operationalize complex, industry-specific tasks. In the legal sector, Abdurahman et al. develop a framework called Lex2KG that automatically converts unstructured PDF documents (such as legal statutes) into structured KGs, enabling advanced applications such as SPARQL querying and automated legal chatbots [2]. Guo et al. researched KGs in network infrastructure, proposing an intermediate KG representation, NetKG, that models routing protocols to automatically synthesize configurations and verify the consistency between high-level intents and generated outcomes [45]. Similarly, within software engineering, parsing code repositories into KGs provides structured contextual retrieval, significantly enhancing the output of large language models (LLMs) during repository-level code generation by maintaining stylistic consistency and minimizing duplication [11]. Other specialized variations include hierarchical KGs for modeling complex legal structures [93] and temporal KGs for reasoning over time-bound information [59].

To address the bottleneck of manual graph construction and scale these representations, recent research increasingly focuses on unifying LLMs and KGs. Pan et al. outline a roadmap for this integration, emphasizing the inherent complementarity of LLMs and KGs: LLMs offer exceptional generalizability but are black-box models prone to hallucination, whereas KGs provide factual accuracy and structural interpretability but struggle with unstructured text processing [76]. Within the "LLM-augmented KG" framework, LLMs serve as powerful text encoders and relation extractors to automate end-to-end graph construction [76]. This methodology has proven highly effective in specialized domains, as Zhao et al. constructed a framework to parse various research papers into a KG about the circular economy [109]. Huang, Yan, & Zhang automated knowledge extraction from marine accident reports and proposed an LLM-based method for generating KGs. Because these marine accidents are compliance-heavy environments and cannot tolerate hallucinations, a dual-perspective quality assessment framework was also proposed to evaluate structural complexity and semantic accuracy [53].

Despite this proficiency, generating structured graph data directly remains a persistent challenge, as LLMs often omit triples or fabricate relations during text-to-graph generation. To combat this, Han et al. proposed an iterative framework called Prompting with Iterative Verification (PiVe). It employs smaller, fine-tuned verifier models to cross-reference the LLM's semantic graph against the source text, dynamically appending corrective instructions to the prompt to ensure the structural output strictly aligns with the source information [47].

2.3 Research Gaps

The research above shows that automated, machine-readable governance in decentralized architectures is highly desired and increasingly feasible. Several challenges still remain unaddressed in the literature discussed above:

- **Manual translation of human-readable contracts into executable rules:** The literature demonstrates that federated computational data governance is essential for scaling decentralized data architectures, mostly with an emphasis on automated enforcement and policy-as-code. However, the practical implementations still require manual labour to translate human-readable agreements into executable rules.
- **Heterogeneity and lack of standardization of data contracts:** While data contracts are essential for defining the technical, operational, and regulatory boundaries of distributed architectures, their heterogeneous nature complicates the creation of automated enforcement.
- **Lack of practitioner-grounded data contract literature:** The academic literature on data contracts remains sparse and largely conceptual, deriving its definitions and components from written sources (white and gray literature) rather than from how practitioners design, negotiate, and apply data contracts within active data mesh implementations.
- **Lack of deterministic LLM-based text-to-graph generation:** Although LLMs can alleviate the bottleneck of manually constructing a graph, they frequently omit triples or fabricate relations. Existing research, such as PiVe, reduces these errors through iterative verification using additional fine-tuned models, but this adds complexity. These solutions do not guarantee the deterministic output that governance requires.
- **Domain-bound knowledge graph construction:** Existing methods that construct KGs from text are often developed and validated within individual, narrow domains, such as the legal domain with Lex2KG or marine accident reports. Given the recognition that one-size-fits-all ontologies are insufficient for specialized fields, these methods do not

transfer directly to the heterogeneous domain of data contracts, to which they have not yet been applied.

This thesis addresses these gaps by using LLMs to extract heterogeneous data contracts into a structured KG through a reproducible extraction process, producing a machine-readable intermediary model that serves as a deterministic foundation for automated contract enforcement. This research applies KG construction to the previously unaddressed domain of data contracts and grounds the work in practitioners' perspectives, complementing the academic literature.

Chapter 3

Research Design

This research is guided by the Design Science Research (DSR) methodology. To ensure a structured execution and rigorous quality assurance, this research uses two foundational DSR frameworks: the conceptual framework proposed by Hevner et al. [49] and the methodology process model developed by Peffers et al. [79]. These two frameworks work together to guide this research. While the process model (illustrated in Figure 3.1) provides a six-step roadmap for research execution, the conceptual framework (illustrated in Figure 3.2) establishes the criteria for rigor and defines the type of IT artifact to be created.

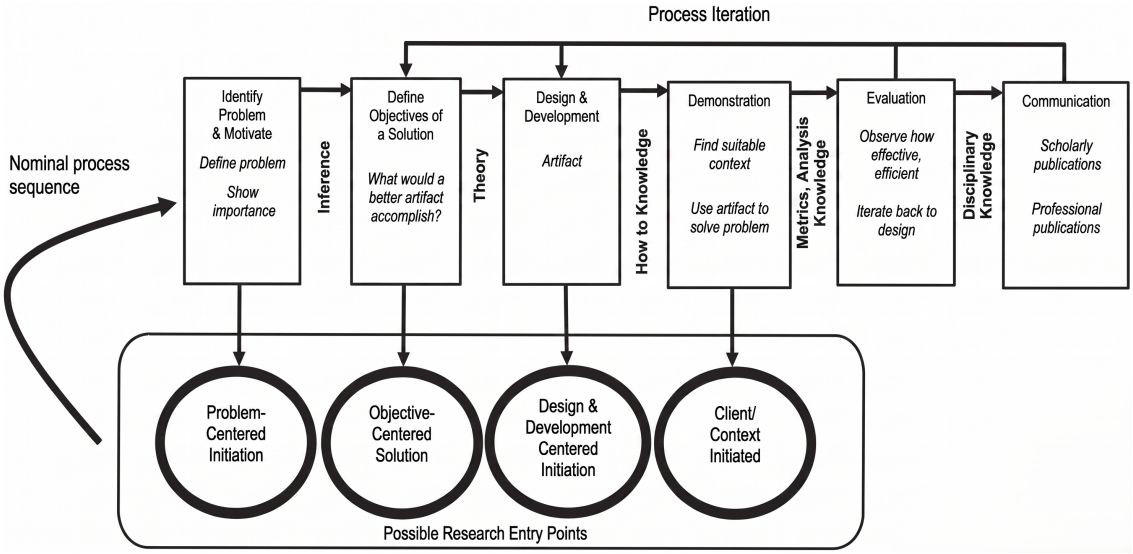

Figure 3.1: DSR methodology process model [79]

Before choosing DSR, action research was considered as an alternative framework, since the two share several features. Action research also works with an intervention methodology in which the researcher collaborates directly with practitioners to employ an interventionist approach to address a real organizational problem. Research using this methodology solves problems through iterative cycles of diagnosing, action planning, action taking, evaluating, and specifying learning [90]. In principle, such an approach could suit a practice-driven problem like data contract enforcement. When the objective is to describe, explain, or predict a phenomenon, action research and case study research are the appropriate methods; when the objective is to design and develop artifacts and prescriptive solutions in a real or simulated environment, DSR is the most suitable method [28]. Action research is primarily oriented toward organizational development, generating knowledge by intervening in how an organization operates. The contribution of this research, by contrast, is a designed artifact, evaluated through technical and user-based evaluations. DSR is therefore the better fit: the main research question, how LLMs can help enforce data contracts, together with its sub-questions on the architectural principles (**SRQ3**) and components (**SRQ4**) required to do so,

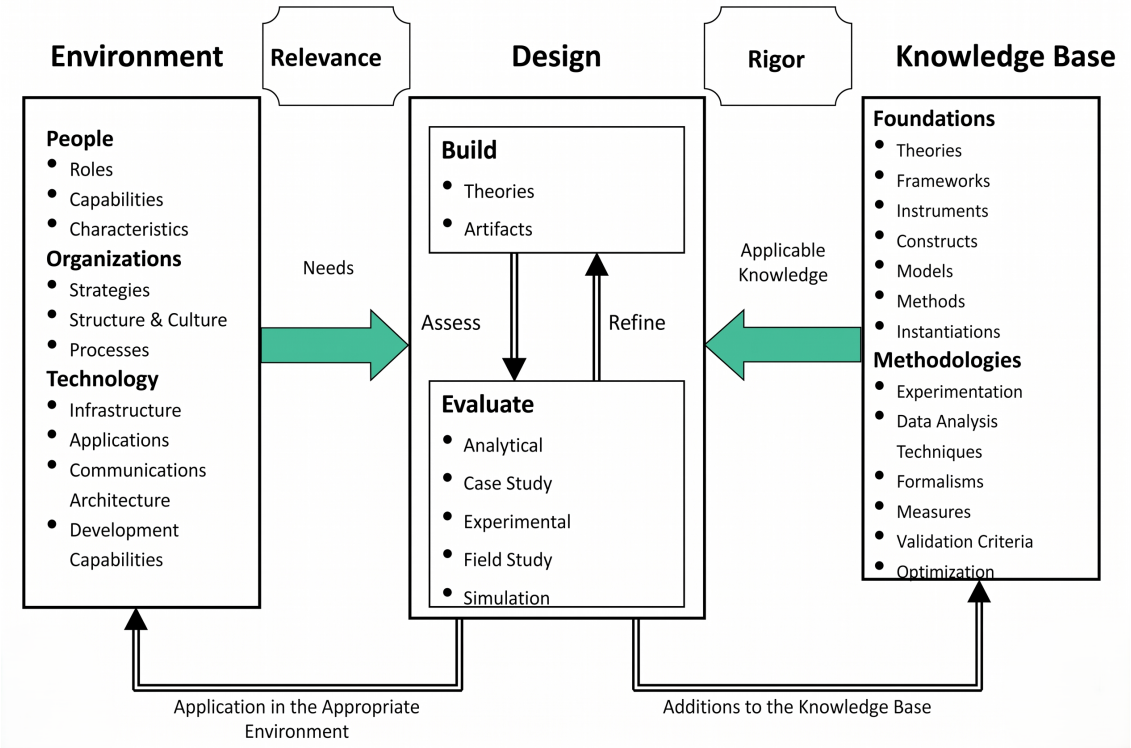

Figure 3.2: Design science research framework [49]

are design questions that ask how to construct a working solution rather than to describe an existing phenomenon. DSR is oriented toward the construction and evaluation of artifacts that solve real-world problems and therefore better matches the research design.

Within this methodology, it is essential to define the dual objectives of relevance and rigor that drive DSR. Relevance ensures that the research is motivated by business needs and addresses the correct problems within a specified environment. This environment outlines the problem space, including the people, organizations, and existing technologies [49]. Rigor, on the other hand, guarantees that the research is grounded in existing scientific foundations and methodologies. It demands that the appropriate application is used to guide the design process, as well as the use of proven empirical and analytical methods to evaluate the artifact [49, 79]. Together, these paradigms form the foundation of DSR, where relevance justifies the artifact, while rigor ensures the validity of the design and its scientific contribution [49].

In the context of Hevner's artifact classification, this research produces a composite solution. The core contribution is an instantiation [49]. We propose a prototype that operationalizes a novel method for translating diverse data contract formats into a structured KG model. This intermediary model provides the foundation required for future research and autonomous agents to automate data contract enforcement. Because data contracts and their surrounding decentralized architectures have not yet been extensively researched, this project uses a problem-centered initiation, one of the four possible entry points in the six-step DSR framework [101].

The execution of this research strictly follows the six activities defined in the DSR methodology process model [79]:

1. **Problem Identification and Motivation:** While the challenges of decentralized governance are generally recognized [60], starting with problem identification establishes the research motivation and reduces the gap between theory and practice [28, 50]. In this initial phase, we merge qualitative research with DSR by conducting exploratory,

semi-structured interviews with eleven industry experts to gain a more comprehensive understanding of both cause and effect and to enable the practical development of artifacts [32]. Aligning with the DSR objective of creating broadly applicable knowledge [107, 57], these interviews provide an in-depth understanding of how data contracts are implemented in practice and identify the bottleneck of manual translation.

2. **Define the Objectives for a Solution:** Using the insights from the industry experts, the requirements for the artifact are defined. The objective is to create a framework that standardizes these heterogeneous contracts into a unified, machine-readable format.
3. **Design and Development:** This phase translates the requirements into the actual artifact. It involves engineering the semantic ontology for the KG and developing the LLM-assisted extraction pipeline to populate it.
4. **Demonstration:** The developed instantiation is demonstrated by processing a sample set of data contracts, proving its ability to successfully generate the intermediary Knowledge Graph.
5. **Evaluation:** The artifact is rigorously evaluated on its syntactic accuracy using SHACL validation, its semantic fidelity using a partial gold standard, and its practical utility through follow-up assessments with the original 11 industry experts.
6. **Communication:** Finally, the problem, the artifact, and its utility are communicated through this thesis, contributing new disciplinary knowledge to the fields of federated data governance and generative AI.

The remainder of this section details the interview methodology: the design of the interview guide (Section 3.1.1), the participant selection criteria and process (Section 3.1.2), the interview protocol (Section 3.1.3), and the analysis of the resulting qualitative data (Section 3.1.4).

3.1 Problem Identification

In the first phase of the DSR, to bridge the gap between theoretical literature and practical application, semi-structured interviews were selected. This format provides greater flexibility than the rigid nature of structured, questionnaire-based interviews, while still allowing us to effectively guide the dialogue, an advantage over unstructured interviews [68]. By employing a semi-structured design, the interaction becomes more conversational, creating space for negotiation and in-depth discussion with the interviewee.

The qualitative data gathered from the semi-structured interviews were thematically analyzed using a comparative, iterative, and inductive process informed by grounded theory [75]. This allows us to identify and analyze patterns in the interviews using Reflexive Thematic Analysis (RTA) in the initial phase [16] and to formulate valid DSR requirements by integrating grounded theory in the second phase. Defining requirements demands more than summarizing descriptive themes; it requires a grounded, relational thematic structure to accurately map the interactions between the categories and themes found in the interviews. By mapping these relationships, key functional requirements can be identified and translated into Design Principles (DPs) [66]. We integrate grounded theory using three-level thematic abstraction: interview quotes are coded, grouped into categories, and similar categories are aggregated into themes [75]. This is then used to create a narrative and to identify the requirements for our artifact.

The following section details the methodology of each step in the interview process. Section 3.1.1 describes the design of the interview. Following this, Section 3.1.2 explains the criteria and process for participant selection. The interview protocol is discussed in Section 3.1.3. Finally, Section 3.1.4 presents the approach used to analyze the resulting qualitative data.

3.1.1 Interview Questions

Prior to participant recruitment and data collection, the interview questions (or interview guide [3]) were developed. The questions were crafted using a mixed-methods approach, incorporating various question types identified by Mann [68] and sequenced according to established guidelines [3, 41].

Following these guidelines, the interviews began with foundational, easygoing questions about the participants' current roles and years of experience. Then the interview transitioned to the core questions, divided into multiple parts. Each new section began with an introductory or opening question, frequently accompanied by planned follow-up questions. While the interview guide provided the primary structure, additional question types were utilized throughout the conversation. These additional questions included probing questions to elicit more examples, direct and indirect questions to gather contextual background, and interpretive questions to accurately capture the interviewees' perspectives [68, 41].

To bring the session to a natural close, the discussion shifted towards the practical, day-to-day challenges the experts encounter in their roles. The final part involved a question about participants' willingness to participate in a follow-up in the later stages of the research [3].

Recognizing that interview design is an iterative process [3], a pretest interview was conducted before formal data collection. The primary objectives were to evaluate the clarity and sequence of the questions, as well as to estimate the overall duration of the session [3, 4, 96, 12]. This pilot session was conducted under realistic conditions, followed by a debriefing in which the participant provided feedback on question comprehension and the flow of the conversation. Pre-testing this way aligns with cognitive interviewing, where we assess comprehension, decision-making, and responses to questions. This allows the researchers to see how people actually interpret and process the questions and make changes before starting the initial interviews [68, 12]. The final set of questions and the rationale can be seen in Appendix A.

3.1.2 Participants

To ensure relevant qualitative data were gathered, a mix of purposive and convenience sampling was used, as both are non-probability sampling techniques. Purposive sampling is an approach often used in qualitative research to select participants based on specific qualities, knowledge, or experience [4, 19]. Convenience sampling is a type of non-random sampling, where participants are selected based on practical criteria, such as geographical proximity [30]. By using both sampling techniques, we allow the participants to meet the criteria regarding the research objective while accounting for their accessibility and willingness to volunteer.

The participants, detailed in Table 3.1, were recruited through the researchers' professional network and via targeted messages on LinkedIn, email, or online contact forms. An effort was made to select participants with diverse experience levels, geographical locations, and industry domains. This approach ensured a balance between including individuals with extensive subject-matter expertise and those capable of providing diverse perspectives [4]. Following 26 initial messages, eleven individuals agreed to participate. These interviews were subsequently conducted over a two-month period.

3.1.3 Interview Protocol

The initial message provided a professional introduction, outlined the study's purpose, and highlighted the relevance of the candidate's background to the research goals. Once a candidate agreed to participate, an Informed Consent Form was sent. The participants were

ID	Current Role	Company Domain	Country	Experience
P1	CEO & Data Engineer	Consultancy	The Netherlands	5 years
P2	Sr. Director Data, Analytics & AI	Retail	The Netherlands	15 years
P3	Lead Data Architect & AI	Retail	The Netherlands	6 years
P4	Head of Data Engineering	Retail	The Netherlands	26 years
P5	CEO	Consultancy	The Netherlands	17 years
P6	CEO	Enterprise Software	Germany	10 years
P7	Sr. Product Manager	Enterprise Software	USA	30 years
P8	CEO	Consultancy	Canada	25 years
P9	Strategy Advisor for Data & AI	Financial Services	The Netherlands	6 years
P10	Lead of Data Governance	Retail	The Netherlands	6 years
P11	Head of Data Governance	Retail	Germany	10 years

Table 3.1: List of Participants to the Study

asked to sign this document, which outlined their rights, data storage protocols, assurances of confidentiality and anonymity, and a request for permission to record the session.

Depending on the participant's preferences, interviews were scheduled around their availability and location. To accommodate the diverse distribution of industry experts, most sessions were held online via Google Meet or Microsoft Teams. When permitted, on-site interviews were prioritized to enable richer, in-person interactions [4].

During the interviews, the interview guide (detailed in Section 3.1.1) served as a primary framework rather than a rigid script, allowing for conversational flexibility [4]. Recording the interviews enabled the researcher to actively listen and engage with the participant, thereby creating space to formulate effective probing questions [3]. Following each session, the audio recording was securely processed through automated transcription software, and the resulting transcription was manually verified by the researcher to ensure high data accuracy.

3.1.4 Data Analysis

The qualitative data analysis was conducted using a hybrid approach, starting with the iterative phases of RTA [18]. RTA is a flexible, interpretative approach to qualitative data analysis that facilitates the identification of patterns in interviews. It highlights the active role of the researcher in knowledge production, in which codes and themes are organically generated by organizing them around central concepts rather than attempting to 'find' predefined codes. The standard process of this methodology follows the iterative six phases illustrated in Figure 3.3.

However, our data analysis diverges at the final stage because RTA captures and categorizes narrative patterns; it does not capture the causal and contextual relations necessary to formulate DSR requirements. To address the need for relations, the analytical approach transitioned to Thematic Analysis informed by Grounded Theory (TAG) after phase 5 (defining and naming themes) [75]. TAG provides the bridge between the coding procedures of thematic analysis and the explanatory, relational mapping techniques of Grounded Theory. To do this, we use a three-level abstraction (codes → categories → themes). Establishing categories between the codes and themes allows for descriptive detail, like the initial codes, while maintaining the conceptual abstraction of the overarching themes [75]. Because this study seeks to answer practical questions regarding data contracts, an inductive, data-driven approach was adopted. An inductive approach dictates that codes are built from the data, this ensures that the overarching themes emerge directly from the transcript. Avoiding defining codes and themes beforehand mitigates the risk of researcher bias and premature

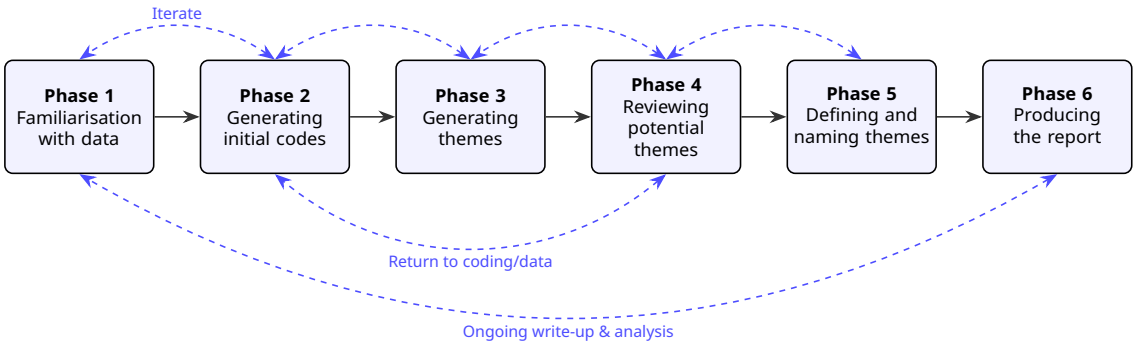

Figure 3.3: The 6 iterative phases of Reflexive Thematic Analysis

closure, ensuring that the resulting insights are grounded in the participant's perspectives [26].

The first step in analyzing the data was familiarizing oneself with it. This involved reading the interview transcripts multiple times before the formal coding started. Following familiarization, the data were systematically coded. This initial coding phase utilized both semantic coding, the explicit information about practices regarding data contracts from the participants, and latent coding, identifying underlying ideas or assumptions not explicitly mentioned [18]. During this process, preliminary codes were assigned to relevant data extracts. As we iteratively reviewed the transcripts, the initial codes were continuously refined: merging specific codes to identify patterns or dividing broad codes to capture more precise nuances. After the first three transcripts, we switched to ATLAS.ti 25.0.2 for Windows [1], a dedicated qualitative analysis software that facilitated better code management and the integration of other useful tools [54].

Once the foundational codes were established, the analysis moved into the grouping phase. Following initial RTA principles, these codes were initially synthesized into candidate themes [18]. However, to align with the TAG methodology and facilitate the relational analysis, the analytical structure was adapted to reflect a three-level abstraction process comprising codes, categories, and themes [75]. The initial candidate codes were iteratively grouped into smaller overarching categories, then merged into broader themes; an example is shown in Figure 3.4. This ensured that the data was organized into well-defined conceptual blocks, such that it could be linked through connections and relationships.

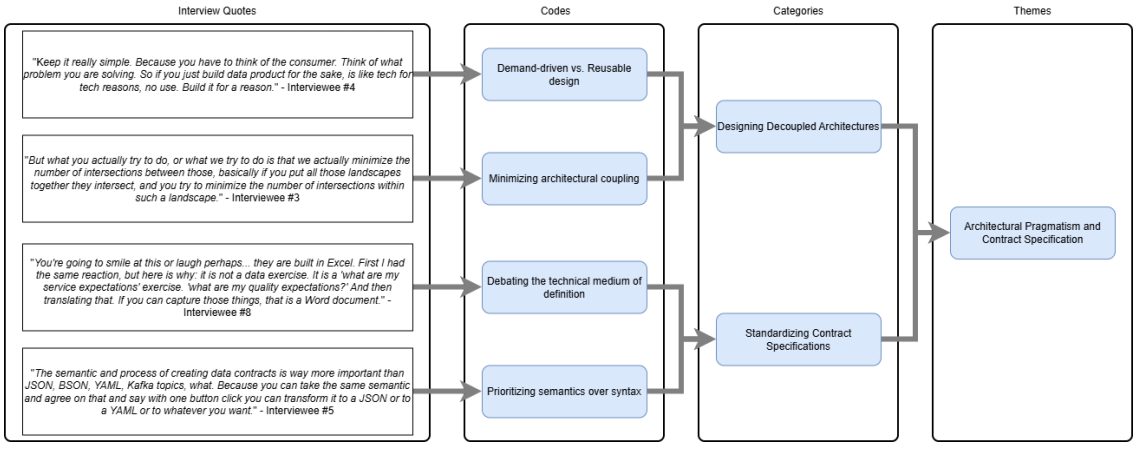

Figure 3.4: Example of thematic analysis on qualitative data

The final step in the process is to transition from descriptive categorization to the structural relations required for DSR artifact development. To achieve this, axial coding was applied, a procedure in which fragmented categories are reassembled for comparison and to establish connections between them [87, 8]. To map the relationships, a paradigm model was utilized (displayed in Figure 3.5). This is a structured, analytical framework to organize the categories into *causal conditions*, a *core phenomenon* (also known as *category*), *context*, *intervening conditions*, *action/interaction strategies*, and *consequences* [8, 63]. Following this relational mapping, selective coding was applied to integrate and refine the emerging theory [8]. In this phase, we identified the central *category* and systematically related all other categories to it [87]. This final analytical step enables us to create an explanatory narrative from the fragmented categories, providing a grounded theoretical structure for identifying the problem and, from there, accurately informing the design of the artifact.

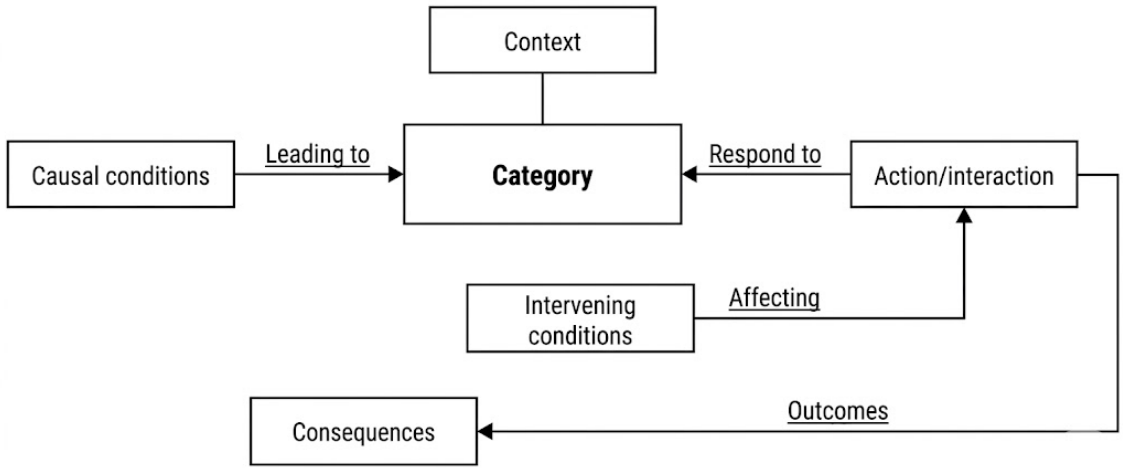

Figure 3.5: Relations between the paradigm model's elements [8]

3.2 Problem Definition

After the final analytical step, we have the theoretical understanding needed to identify the gap between the desirable and current states. In design science methodology, identifying this gap is the primary focus of problem explication, which transforms an initially vague practical issue into a precisely defined and well-justified challenge [57]. In the second phase, where we define the objectives, we compare current user experiences with the envisioned ideal scenario. We use the qualitative data from our interviews to create a central research problem. This step is essential for scoping the project effectively. We hope to ensure that our efforts address the root cause of the problem, thereby reframing the issue as a generic class of problems relevant to broader, global practice [57].

Once the core problem is clearly defined, it serves as the foundation for the deductive and inductive approaches to initiating the design phase. The problem definition allows us to systematically deduce key requirements from stakeholder needs, which are then inductively abstracted into design principles [66]. These principles represent a generic class of solutions and guide the formulation of a clear design problem [66, 107]. By creating these principles, we make a contribution argument, a justified prediction that when introducing our designed artifact into the practical environment, it will successfully bridge the identified gap [107]. Ultimately, establishing this precise problem definition provides a necessary baseline for

continuous evaluation, allowing us to systematically assess whether the final artifact resolves the initial challenges and contributes to both theory and practice [57, 66].

3.3 Artifact Design & Development

The artifact design and development phase bridges the gap between the problem statement and the proposed solution. In this phase, the theoretical understanding and problem definitions are translated into a concrete design and set of requirements, which are subsequently developed into a functional artifact.

3.3.1 Artifact Design

To develop a solution, the design process requires eliciting requirements. A requirement is a desired property of the artifact, formulated to address the previously identified root causes and stakeholder needs [57, 107]. These requirements are categorized into two types: functional and non-functional requirements. Functional requirements specify the functions the artifact must perform within its context [57, 107]. Non-functional requirements address the quality properties of the artifact, describing which non-functional properties it should have [107]. These properties can be internal structural qualities, such as modularity and coherence, or their environmental qualities, such as usability, maintainability, and interoperability [107].

However, since our solution will integrate AI, we must consider the technology's non-deterministic nature; therefore, traditional requirements engineering must be adapted [6]. Specifically, the requirements must be expanded to explicitly include *data requirements*, such as data quality, structure, format, and availability, since data drives system behavior and is replacing traditional code in AI models. Furthermore, the prioritization of non-functional requirements becomes increasingly important. Traditionally important non-functional properties, such as structural properties, may see their relevance decline compared with the growing importance of environmental qualities like fairness, transparency, and explainability [6].

Because non-functional requirements are often subjective, they must be operationalized to ensure the artifact can be evaluated [107]. This process involves translating abstract properties into specific, measurable indicators and establishing acceptance criteria for each indicator. In the context of AI systems, this operationalization must also account for the trade-offs between competing requirements, such as balancing accuracy against explainability or performance versus costs, requiring researchers to establish thresholds [6]. To increase the clarity and utility of the requirements, they are formulated to be atomic, concise, and unambiguous [57]. By establishing these verifiable criteria, the requirements serve a dual purpose: they provide guardrails during the development and form a baseline against which the final solution's quality and success will be validated [57, 107].

After the definition of the specific requirements, the methodology abstracts them into broader categories known as meta-requirements. These meta-requirements characterize the general scope of goals relevant to resolving the identified class of problems [66]. To identify these meta-requirements, we first cluster the functional, non-functional, and data requirements into overarching requirements. By abstracting these low-level requirements, we focus on solving a generalized class of problems and contribute to the broader scientific knowledge [66].

These derived meta-requirements form the basis for the induction of DPs. DPs act as a high-level representation of the class of solutions intended to address the identified class of problems [66]. Unlike standard requirements, DPs are not directly implementable; they describe solutions from a generic perspective. The process of defining DPs involves combining and linking the meta-requirements into emergent principles. These serve as preliminary DPs

that can be refined during further iterations. Because DPs are generalizable and innovative, they serve as a bridge between relevance and rigor in DSR research [66].

3.3.2 Artifact Development

After setting up the requirements, the actual construction of the artifact begins. In Design Science Research, an IT artifact is broadly defined and can take the form of constructs, models, methods, or instantiations [49]. For the purpose of this research, the development phase focuses on creating a functional instantiation, a working prototype that implements the proposed AI-integrated solution within its target environment.

According to Peffers et al., this activity involves determining the artifact's architecture and subsequently creating the actual artifact [79]. The process relies heavily on the previously established functional, non-functional, and data requirements, which act as the blueprint for the system's design. The conceptual models and workflows are translated into a concrete software architecture that defines how the user interfaces, backend logic, and AI models interact to deliver the desired functionality.

Furthermore, the development of an artifact is rarely linear; rather, it is an iterative search process [49]. As the design is translated into a concrete instantiation, continuous refinement is required to navigate the complex problem space and balance competing constraints.

Ultimately, the goal of this phase is to produce an artifact that is practical and can be interacted with in its intended problem context. This resulting instantiation serves as the tangible proof of concept, bridging the gap between theory and practice, and setting the stage for the subsequent demonstration and evaluation phases [79, 107].

3.4 Artifact Demonstration

Following the design and development phase, the fourth activity in the DSR methodology is the demonstration of the artifact [79]. According to this framework, the primary objective of the demonstration phase is to provide a practical proof of concept, establishing that the developed instantiation can effectively address one or more instances of the identified problem. In the context of this research, this demonstrates that the proposed approach can successfully automate the translation of natural-language data contracts into a formalized structure.

Methodologically, this demonstration is conducted by applying the developed artifact to a representative sample of existing data contracts. These contracts are selected to reflect the typical structural complexities and varying business rules found within decentralized data environments. The demonstration acts as a controlled execution, observing the artifact's operational behavior as it processes unstructured inputs to yield the defined intermediary model.

The focus of this phase is strictly on establishing functional feasibility rather than architectural specifics or performance metrics. By successfully executing this applied scenario, the demonstration confirms that the theoretical design can be operationalized. This step is a critical prerequisite, ensuring the artifact is functionally sound before advancing to the rigorous assessment in the following evaluation phase.

3.5 Artifact Evaluation

The fifth phase of the methodology involves a comprehensive evaluation of the developed artifact to ensure its technical correctness, structural efficiency, and practical utility. To assess the quality of the generated KG, we will use an intrinsic and extrinsic evaluation approach.

Intrinsic evaluation looks at the inner workings of the KG. It provides numerous quantitative metrics for assessing the output's quality and performance [20].

3.5.1 Intrinsic Evaluation

To evaluate the KG intrinsically, we will employ a partial gold standard approach [78]. This process involves manually labeling a representative subset of entities and relations to establish a ground truth of correct axioms. The artifact's completion quality will then be measured against this standard using precision, recall, and F1-Score. These metrics are suitable here because the research question concerns whether the framework extracts the correct graph elements without omission: precision captures the correctness of what is extracted, recall captures the completeness of the extraction, and the F1-score balances the two into a single measure of overall extraction quality [20]. The necessity of quantitative validation is underscored by the fact that even mature, widely used KGs are prone to errors. For instance, it has been reported that approximately 20% of the interlinks between DBpedia and Freebase are incorrect [78].

We calculate these traditional metrics by defining True Positives (TP), False Positives (FP), and False Negatives (FN) within the context of the extraction process. TP represents the number of correctly extracted nodes and relations. FP refers to the number of extracted triples that do not exist in the ground truth, while FN indicates the valid triples present in the ground truth that the system failed to extract.

Because LLMs may express the same triple with minor variations (e.g., a differing identifier, or using snake_case instead of CamelCase), each triple is first normalized and then matched against the gold standard. A triple is counted as a TP when it is identical to a gold standard triple (exact match) or if its text similarity to a gold standard triple is at least 90% (fuzzy match). The similarity between two triples is measured using the Indel similarityz, also known as the Longest Common Subsequence (LCS) distance [102]. We adopt this measure because it is robust to the insertions and deletions in the variations in LLM-generated output, such as added or omitted tokens.

Using these values, precision and recall are calculated to measure exactness and completeness, respectively:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

To provide a single, balanced evaluation metric of the performance, the F1-score is computed as the mean of precision and recall:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.5.1.1 Efficiency

Beyond the accuracy of a single extraction, we report two efficiency indicators for each configuration: the amount of tokens consumed per extraction loop and the execution time. These are included because the pipeline is intended to ingest many contracts across an organization, so per-extraction cost and latency partly determine its practical deployability. They are also diagnostic rather than purely operational: because the verifier re-runs generations that fail syntax or SHACL validation, the token and time cost of a strategy reflects its structural reliability and not only its prompt length, so a strategy that appears cheap per call can become expensive once its retries are counted. Token usage is tied to the model's

tokenizer and is therefore interpreted as a relative comparison between configurations rather than an absolute cost that transfers to other models or vendors.

3.5.1.2 Determinism

The traditional metrics defined above measure the quality of a single generated KG against the gold standard. However, because the extraction pipeline relies on an LLM, identical inputs are not guaranteed to produce identical outputs. LLMs are non-deterministic by nature, so the same prompt can yield different responses on separate requests [74].

To assess determinism, we run the proposed pipeline a fixed number of times under identical settings and quantify the differences of the resulting metrics and generated output. As the principal measure of dispersion, we adopt the coefficient of variation (CV), defined as the ratio of the standard deviation (σ) to the mean (μ) of a set of repeated measurements and is often reported as a percentage [84]:

$$\mathrm{CV} = \sigma / \mu$$

CV originates in measurement science where it measures how much a result moves when the same sample is measured again under unchanged conditions [84]. We select the CV because it is dimensionless, which allows variability to be compared fairly with different scales [84]. Our metrics range from triple count in the order of hundreds, while the traditional metrics are bounded in $[0, 1]$.

While the CV captures the dispersion of a scalar metric, it does not reveal whether repeated runs produce the same triples. To quantify this, we measure the overlap between the triple sets of the repeated runs using the Jaccard index [34]. This index is defined for sets A and B as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|},$$

where A and B are the sets of triples generated by the LACE Framework. In the formula, $|A \cap B|$ counts the triples that appear in both runs, while the denominator ($|A \cup B|$) counts every distinct triple produces by both runs. Their ratio is the proportion of all observed triples that exist in both outputs. When there are no shared triples, the index equals 0 and when both runs produce exactly the same triples, it is 1. This index is reported in two-fold: exact matches, in which two triples are considered equal only if their normalized strings are identical, and fuzzy matches, in which they are considered equal when their text similarity is at least 90%, using the Indel similarity. The exact index reflects one-on-one reproducibility, while the fuzzy index reflects semantic reproducibility.

3.5.1.3 Graph Entropy

In addition to the exactness and completeness measured by precision, recall, and the F1-score, we also evaluate the complexity of the generated KG. To achieve this, we employ the graph information entropy metric proposed by Huang et al. [53], which quantifies structural disorder and information dispersion based on the probability distribution of node degrees. First, we calculate the total degree (the sum of in-degree and out-degree) for each node. Let d_i represent the degree of node i. The normalized degree, p_i, is calculated as the ratio of the node's degree to the sum of all node degrees within the graph:

$$p_i = \frac{d_i}{\sum_j d_j}$$

The graph information entropy, H, is then computed using the probability distribution of these normalized degrees:

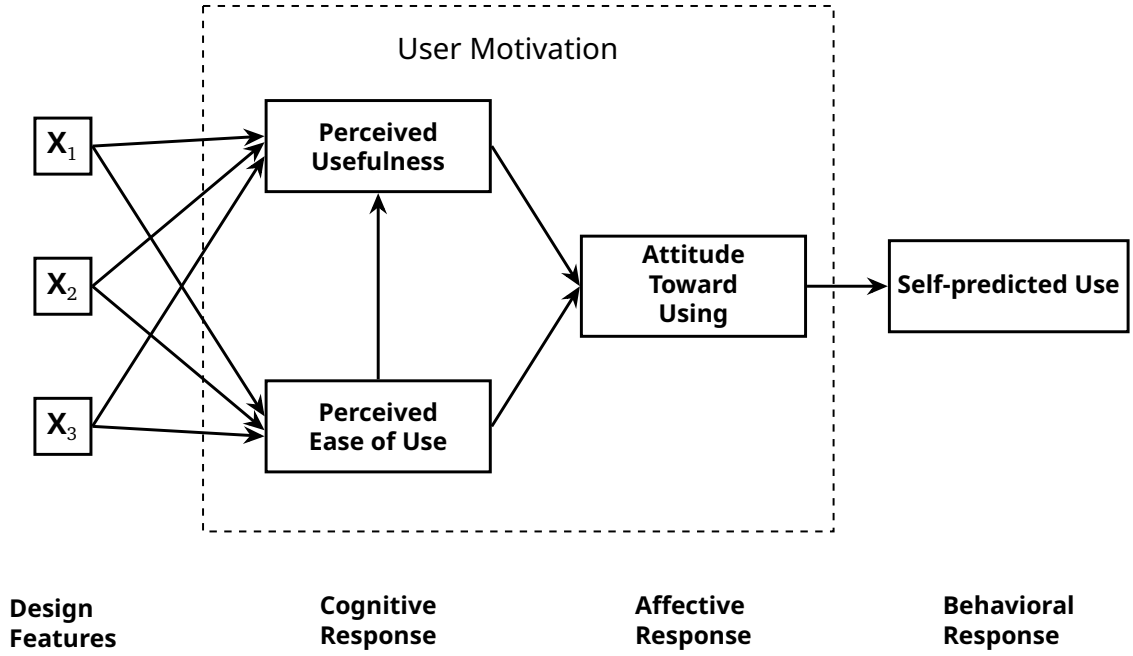

Figure 3.6: Original Technology Acceptance Model (TAM) causal relationships proposed by Davis (1985) [23]

$$H = -\sum_{i}^{N} p_i \log_2(p_i)$$

A higher entropy value indicates a more complex graph structure, reflecting a higher density of information across nodes and relations. Furthermore, to determine whether the graph's complexity effectively contributes to its core utility, we define a main chain that includes only the graph's essential relations, that is, the relations that encode the core data product and contract governance logic (lineage, ownership, ports, and policy constraints) rather than descriptive metadata. We then compute a complexity ratio by dividing the graph information entropy of this isolated main chain by the entropy of the overall knowledge graph:

$$\text{Entropy Ratio} = \frac{H_{\text{main\_chain}}}{H_{\text{full\_graph}}}$$

This ratio assesses whether the structural complexity required to describe the core data product and contract logic dominates the network, or whether many other elements contribute to the complexity [53].

3.5.2 Extrinsic Evaluation

Finally, an extrinsic evaluation will test the artifact's practical implications, and its perceived usefulness, ease of use, and users' attitudes towards using it will be assessed. This will be achieved by conducting a user study based on the Technology Acceptance Model (TAM) as defined by Davis [23]. This framework is among the most widely used for analyzing and explaining user adoption of technological innovations [9]. By gathering structured feedback from end-users, this final evaluation step provides a holistic assessment, ensuring the artifact is not only technically robust but also valuable and accessible to its intended audience.

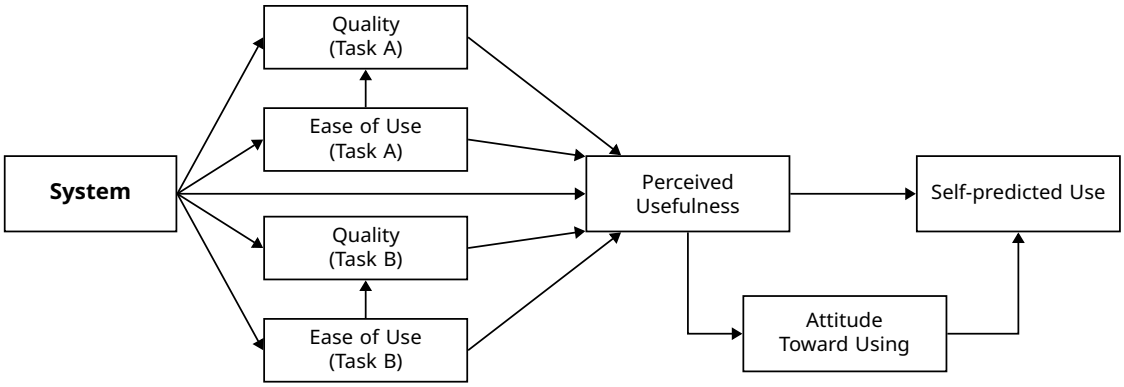

Figure 3.7: TAM3 hypothesized relationships presented by Davis (1985) [23]

To establish the theoretical foundation for this evaluation, it is necessary to define the core constructs of the TAM that will be assessed. The causal relations between the constructs will follow the original model proposed by Davis [23] and are shown in Figure 3.6. By defining the relations, we can estimate how each variable influences a user's final decision. The core constructs of the TAM are defined as:

- **Perceived Usefulness (USEF):** USEF is defined as "the degree to which an individual believes that adopting a system will enhance their job performance" [22]. Because organizations reward better performance, users are driven by these functional benefits, making USEF the most dominant factor of system acceptance [22].
- **Perceived Ease of Use (EOU):** This variable measures the extent to which a user expects their interaction with the system to be completely free of cognitive effort [22]. Empirical evidence suggests that EOU acts as a causal antecedent to USEF, meaning that easier systems allow users to allocate more effort to their actual work, thereby increasing overall usefulness [22].
- **Attitude Toward Using (ATT):** ATT reflects the degree of evaluative affect linked to an individual's use of the system on the job [23]. It serves as a metric that reflects the user's underlying motivation and feelings toward adopting the technology [23].
- **Actual System Use:** Representing the behavioral criterion of the TAM framework, this construct refers to an individual's actual usage of the system in the context of their job [23].

As this research will result in an intermediary model, different tasks can be performed by the output. To evaluate where this model generates value and to whom, we use the TAM3 extension, which extends the original TAM with a Perceived Output Quality (QUAL) and elaborates the EOU and QUAL constructs on a task-specific basis rather than as generalized perceptions [23]. The causal relations that come from the TAM3 model are shown in Figure 3.7, also proposed by Davis [23]. Because the output's utilization varies significantly across use cases, a generalized evaluation can obscure specific details. TAM3 accounts for this variance by introducing a moderating variable: The importance (IMPORT) of a specific task to a user's job.

The overall USEF of the system is determined by the perceived costs (measured by EOU) and benefits (measured by QUAL) relative to a specific task domain, weighted by how important that task is to the individual [23]. Davis models this interaction between the system's capabilities and the user's job requirements using the following equation:

$$USEF_{system} = \Sigma_{task}(BENEFIT_{task,system} - COST_{task,system}) * IMPORT_{task}$$

By isolating tasks in the evaluation and weighing them against the job requirements, the TAM3 framework allows for a highly granular analysis of which features drive user adoption and for which target demographics. To apply this, we define a set of questions for each construct. The participants will perform a task and then answer the questions for each construct using a 5-point Likert scale, with 1 representing *Strongly Agree* and 5 representing *Strongly Disagree* [31, 23]. We will subsequently validate the reliability of this questionnaire using Cronbach's alpha to ensure the data is reliable [23].

Chapter 4

Problem Exploration & Identification

This section presents the results of the exploratory phase of the problem identification DSR cycle. We will first present the qualitative research results in Section 4.1 through the thematic map and axial coding paradigm. The findings of the qualitative results will then be summarized in Section 4.2. Based on these results, we will provide a definition of a data contract and compare it to the definition of the academic literature in Section 4.3. The identification of the problem will be outlined in Section 4.4. The objective and requirements of the artifact are described in Section 4.5.

4.1 Interviews

Since DSR is highly problem-centered, the context surrounding the exploratory phase is driven by practice. To systematically assess the complexities and immaturity of data contracts, the interview findings are presented narratively. While the initial thematic analysis identified five themes in the data, a comprehensive thematic map, along with the complete list of codes, categories, and themes, is provided in Appendix B. Presenting these findings as descriptive themes fails to capture their dynamics.

Therefore, this chapter is structured by following the relational flow of the axial coding paradigm as depicted in Figure 4.1. By mapping the causal conditions, contexts, action strategies, and consequences surrounding the core phenomenon, data contracts, this section establishes the theoretical foundation required to inform the artifact design.

4.1.1 Causal Conditions

In the axial coding paradigm, causal conditions represent the foundational events, circumstances, or pressures that trigger the emergence of the core phenomenon [87]. The data revealed that the transition toward decentralized data architectures is not only a technological trend but a response to organizational constraints. The analysis reveals that the ongoing challenge of managing distributed data ultimately drives the formalization of data interactions. This section details how the friction between centralized control and local speed establishes the necessary environment for data contracts to become structurally central.

4.1.1.1 Tension in Federated Governance

The findings suggest that the primary causal condition driving the centrality of data contracts is the tension between the domain and the federated governance models, a theme extensively discussed across all participants. Organizations face a continuous struggle to balance the competing desires for centralized efficiency and localized, domain-driven agility. The analysis reveals that navigating this balance requires moving away from a legacy of centralized policing while actively preventing architectural fragmentation.

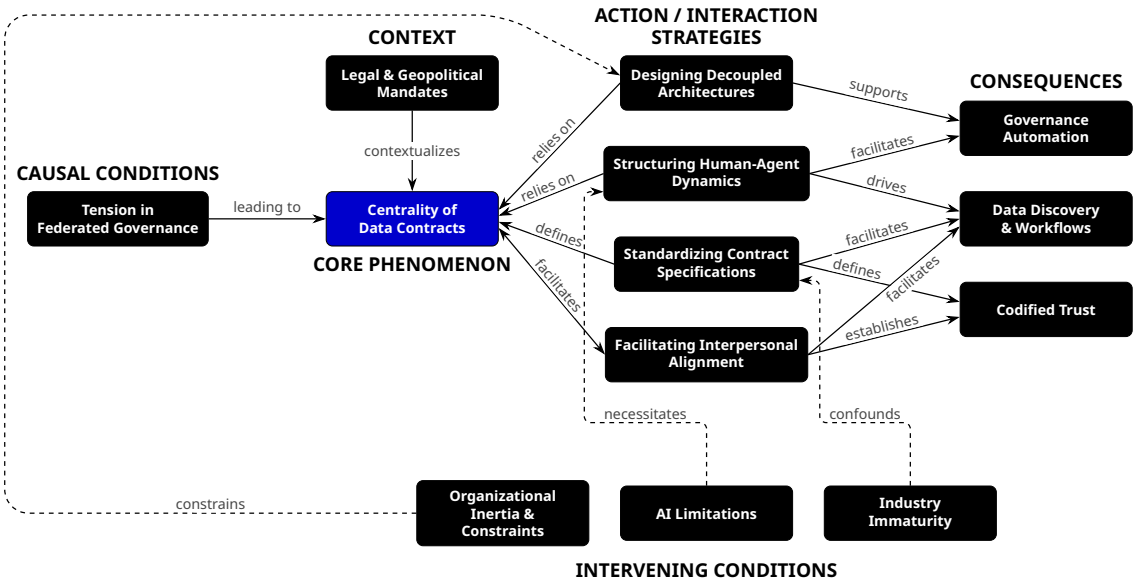

Figure 4.1: Coding Analysis of the Phenomenon [8, 87]

> P8: *"there is a constant balance between cost management, optimizing for cost, and optimizing for speed. If you're optimizing for cost, you're going to centralize your capabilities and leverage the resources, today still relatively expensive resources, so that you can distribute that capability evenly across the enterprise.* [...] *if you're optimizing for agility, then you're doing something very different. If you're optimizing for agility, you want to put the data as close to the people who can make use of it, perhaps by delivering products faster, optimizing the customer experience, or meeting regulatory requirements. But you want to move the data as close to the people who can make the decisions, the right decisions, quickly. So that's a completely federated model."*

> P11: *"Regarding the tension, it's not just a balance, it's a struggle. Our department is part of a large division, and we also have other brands that belong to us. These big companies are doing stuff on their own behalf. They have a high degree of autonomy in how they do things from a business perspective. Today, we want to merge it through omnichannel thinking.* [...] *We want to have the data contracts in one central place, especially to allow cross-referencing with others and reusing business descriptions and stuff like that. But the responsibility itself is federated. Otherwise, the organization and culture are too complex."*

The data illustrate that traditional, centralized data management approaches inevitably become operational bottlenecks. Centralizing capabilities initially served to optimize costs, but this compromised speed and agility. It demonstrated that centralizing data teams forces a small group of people to comprehend various business contexts. This creates a high degree of organizational dependency, making it impossible to keep up with the business demands. Furthermore, when central data or governance teams attempt to dictate a canonical data definition, it ends in friction and is often met with resistance from the business units.

> P10: *"What we saw was that there was a high degree of organizational dependency on a very small group of people. And those people had to understand exactly what the data is, what it does, what can be done with it, and what can't be done with it to make it ready for the business. That went very slowly, very laboriously."*

> P7: *"I've seen data governance teams that were really big, and they wanted to take control of how the data was managed at the business unit level, but in a very end-to-end way. What I mean by that is trying to get real control of the definition of the data and all these things. And they're not successful because they are not welcomed by the business unit."*

To overcome these bottlenecks and accelerate value creation, organizations must adopt domain-driven, federated models. The data shows that, to optimize for agility, organizations must place data and its governance as close as possible to the business and to the people who can make informed decisions. In this federated structure, central governance is relegated to defining regulatory guardrails, while local domains are granted the autonomy to build and govern data products tailored to their specific operations.

> P7: *"Your platform and the governance principles that you set up need to be so good that they can just plug and play. Or they have to onboard skills on their team to be able to understand it and then you also can move ahead. I like that idea more because then innovation also comes from the different domains, who understand their use cases and they are able to drive it."*

> P9: *"Governance sets the requirements. They determine what it must comply with and who is responsible. Architecture then translates that into standards for building, such as platform and technology standards and data product standards, et cetera, et cetera. The data domains do it; they are responsible for execution and must demonstrate that they work in accordance with the standards and policies we establish."*

The analysis revealed that granting this local autonomy comes with a vulnerability: cross-domain and handover friction. When independent business units operate as isolated data producers, the lack of tight centralized oversight frequently breaks downstream dependencies. The data indicates that producers often alter schemas or data structures without notifying their data consumers. Without structural mechanisms, local autonomy quickly degrades into a fragmented and ungoverned ecosystem of isolated data silos.

> P1: *"People who produce data must be aware that their data is used elsewhere and then at least give a signal. So it is a human problem. Namely: just communicate to your colleagues that something changes. But first of all you have to be aware of: okay, I have downstream dependencies."*

> P4: *"The problem is things change, people don't realize that it needs to be updated. It is okay that things change, but they don't update a data contract. Then what happens is they expect something, something else comes and this is where it goes wrong."*

This constant balancing act forms the core causal tension. The necessity to resolve this friction without reverting to centralized architectures directly leads to the core phenomenon: data contracts. The findings indicate that data contracts emerge as the required socio-technical bridge, serving not as technical enforcement, but primarily to align disconnected domains. By providing a structured framework for discussion, they align business and technical stakeholders to establish trust across a distributed landscape.

4.1.2 Core Phenomenon

At the center of the axial coding paradigm lies the core phenomenon, the central point that links all other categories into an explanatory whole. The data identifies *Centrality of Data Contracts* as this structural component. It is not just a reaction to decentralized architectures but a definitive mechanism that enables them to operate.

4.1.2.1 Centrality of Data Contracts

The data suggest that the ability to execute distributed data architectures is entirely dependent on establishing these data contracts, which have been discussed by all participants. The participants illustrate that without a formal descriptor outlining the behavior, ports, and expectations of a data product, the decentralized model risks complete failure.

> P4: *"I think that if you don't have a data contract in place you are set up for failure. Complete failure. I think everything comes down to a data contract."*

> P6: *"I think data contracts are essential, or the key to do governance in an organization."*

> P11: *"As we move to Google platforms, the rule is clear: before any data point flows, there must be a data contract.* [...] *The data contract is really the source of truth for every single metadata point: responsibility, SLAs, and business descriptions."*

Beyond the implementation, the contracts act as a clear dividing line, shifting the organizational approach from treating data as an asset to managing it as a product with defined boundaries and ownership. The analysis indicates that this division is essential for realizing decentralized design, as it forces data producers to specify what they offer to the organization.

> P3: *"From data governance it is looked at: that is this domain, or this subdomain is attached to that and this owner is attached to it. And basically with such an owner, they can then actually give approval on that data product. So that runs via a data contract."*

> P5: *"I think it* [data contracts] *is essential for data mesh. It is essential for any demarcation of creating these boundaries between product thinking, and "this is my product, that is your product", if you really want to step away from asset thinking to product thinking."*

Furthermore, the data contracts serve as the primary mechanism for establishing trust and reliability as data goes through complex architectures. Most often, downstream dependencies break when upstream sources change unnoticed. By establishing a formalized agreement at every layer, organizations prevent unexpected breakages. This emphasizes that a contract enforces exactly what is promised, therefore, codifying the operation of data across domain boundaries.

> P8: *"Anytime data moves from one place to another, whether it's through a pipeline, a batch job, an API, an event stream, there needs to be a way to get that trust, that confidence that what was sent in and transformed is actually appropriate. The data contract provides that ability."*

> P2: *"But once those data contracts start ensuring that we have controlled changes within our data landscape, and data always lives, it changes and it goes cross-domain through a landscape; I always say "Data doesn't respect your domain boundaries" because it goes through everything as that sales order suddenly pops up at Finance again, and then suddenly again in the commercial domain. So data contracts ensure that we are in control of that and that it is reliable, that increases consumer trust."*

Building on trust and reliability, the participants noted that data contracts are central, serving as a standardized, universal interface. The findings suggest that the actual underlying technology becomes irrelevant when a contract is applied. By acting as an official interface, a data contract allows organizations to enforce uniform governance protocols across a fragmented landscape.

> P6: *"A data contract is basically just protecting a dataset that you have as an API for others. As a read-only, SQL-based accessible, or not SQL-based accessible. So you can have a data contract for a CSV file on SFTP. It's totally valid. And you can still use all of these benefits from there. An Excel file on a Windows file share can also have a data contract. There's nothing wrong about that. It's the same, you're sharing data and you guarantee something, you make it an official interface."*

> P7: *"We started doing data products and we realized that we needed something as a descriptor of the data product, describing both the data product and the way the data was behaving. Because a data product has potentially many ways to interact with the data through output ports."*

Finally, the centrality of data contracts extends beyond current human-to-human architectures, becoming the essential metadata layer for organizational data discovery and AI workflows. As we will discuss in later sections (Section 4.1.4.2 and Section 4.1.6.2), data contracts facilitate automated access requests and agentic workflows through their standardized, often machine-readable descriptors of data and quality guarantees.

> P6: *"The last part is: it's essential for AI. Because what does AI need? It needs data. And it needs to quickly find that data. So it does this discovery part, checks whether the data actually would make sense for it, and then it needs to request access and actually use the data.* [...] *But based on the contract, an AI can make a decision: is this data offering helpful for my current task? That's why this data contract is such a huge sphere, because it's so central in so many places."*

Ultimately, the findings confirm that data contracts are the essential foundation of a decentralized data architecture. By enforcing product boundaries, ensuring cross-domain reliability, abstracting technical complexity, and providing a structural basis for automated workflows, they fully operationalize the architecture.

4.1.3 Context

The context in axial coding represents the specific set of conditions or the broader environment in which the core phenomenon operates. The data reveal that the environment in which the transition toward distributed architectures and the centrality of data contracts occur does not arise on its own but is heavily contextualized by external pressures. The analysis indicates that strict legal, regulatory, and geopolitical mandates shape the operational reality for organizations, dictating the boundaries of data sharing and automation.

4.1.3.1 Legal & Geopolitical Mandates

The interviews revealed that legal and geopolitical mandates form the fundamental context, as seen in Figure 4.1, within which decentralized data architectures and data contracts operate (discussed by P3, P4, P6, P8, P9, P10, and P11). Participants indicate that their company is subject to various regulations and cannot independently define its governance. For example, the GDPR in Europe dictates strict privacy expectations, while financial reporting regulations demand transparency and therefore necessitate federated governance models adapted to that region. The findings suggest that even when internal teams wish to operate with greater speed and agility, the regulated guardrails remain the primary context in how data is managed and shared.

> P8: *"Every one of those regions has a different regulatory framework. In Europe there's GDPR, for example.* [...] *Yes, they have some privacy stuff, but there are very different expectations around privacy between different regions.* [...] *So the question is, how do you optimize that? Well, you have to federate the governance such that you understand the unique requirements of that jurisdiction that you operate in."*

> P9: *"The whole reason we do data management is because we have to show De Nederlandsche Bank and the European Bank that we pick out fraudsters, do not participate in financing terrorism, and that we can prove we have enough money in the bank to absorb setbacks.* [...] *We would like to be a bit faster and more value-driven internally, but financial services are, for good reason, strictly regulated. That is always problem number one."*

The data further illustrates that operating within this strict regulatory context fundamentally dictates how data contracts can be structured and enforced. To maintain legal compliance with frameworks like the GDPR and Basel legislation[1], interviewees emphasized that human beings must remain explicitly accountable for governance decisions. The analysis highlighted that concepts, such as "purpose-binding", cannot be fully automated. When data access is requested, the data contract or its interface must be human-readable so that multiple people can understand and approve its use for the newly requested purpose. Consequently, this legal and regulatory requirement demands a human-in-the-loop, thereby limiting the extent to which governance tasks can be handed over to an automated system.

> P3: *"I think from Legal there are naturally certain guardrails that you have to provide. PII and confidentiality. So I think you can say that certain conditions must be met. The entities within it must have a certain classification."*

> P10: *"If I want to consume data, then I first have to fill in a form and describe what I want to do with that data, what I want to achieve with that data. If I want to use personal data, PII data, for my application, I also have to demonstrate that I have permission from Legal."*

> P11: *"We have to keep in mind legal obligations. IT security stuff is mandatory for any department, and the legal processes for GDPR-relevant data are also super strict.* [...] *There is a fear of audits, so it's a clear mandate. During internal or external audits, you just have to cross your fingers that you meet expectations."*

Finally, the findings demonstrated that the context of data governance is also heavily shaped by geopolitical and infrastructure mandates. Participants reported facing concerns about data sovereignty, particularly due to their reliance on foreign technology companies for computing and AI models. The primary risk identified by participants is the potential for confidential customer or enterprise data to escape the organization's environment. Ultimately, the findings indicate that these geopolitical constraints create an environment in which data sovereignty is a non-negotiable prerequisite. This further contextualizes the requirements that data contracts must codify.

> P6: *"It's this movement towards data sovereignty. Because they want to retain control. And at the moment, you don't want to give control to a US company. That's essentially the hard part."*

In essence, the findings show that legal and geopolitical mandates establish the boundaries within which data contracts must function. By enforcing regional compliance, such as human accountability or strict sovereignty requirements, this context shapes how data contracts are structured and limits the degree to which these contracts can be used for automated workflows.

4.1.4 Interaction Strategies

In the axial coding paradigm, interaction strategies represent the steps that actors take to manage the core phenomenon within its context. The analysis identified four primary

[1] https://www.bis.org/bcbs/basel3.htm

strategies utilized by the experts to govern this ecosystem. First, to safely manage cross-domain dependencies, organizations focus on designing decoupled architectures and using data contracts as a protective interface. As automation scales, teams are structuring human-agent dynamics by positioning AI to increase people's productivity. Because contracts must be understandable to multiple stakeholders, organizations face the challenge of standardizing contract specifications, in which semantics are more important than contract format. Finally, to prevent teams from becoming siloed, contracts are employed to facilitate interpersonal alignment by forcing conversations and establishing agreements between data producers and consumers.

4.1.4.1 Designing Decoupled Architectures

Every participant discussed the necessity of designing decentralized architectures. While this consensus aligns with the participants' active involvement in decentralized architectures, the analysis demonstrates that the effort to minimize architectural coupling primarily serves to establish distinct product boundaries and to facilitate decentralized data ownership. Participants emphasized that for data domain-driven design to succeed, teams must be able to operate within their own contexts without being tightly coupled to other domains. However, to safely serve these rigid technical dependencies, teams rely on data contracts. The contract serves as the reliable, agreed-upon interface that enables this operational decoupling, ensuring that downstream consumers remain protected even as upstream source systems change independently.

> P3: *"The first is the system layer, source aligned layer, where we make as few changes as possible* [...] *we have a domain layer. That is basically where we say: we decouple from specific applications.* [...] *And behind that we basically have a consumer aligned layer.* [...] *So one source aligned layer with multiple data products, that is basically an n to m relation, so multiple source aligned layers can feed multiple domain layers. Each of those data products also has its own data contracts.*

> P9: *"I would want multiple data contracts per data product, one per output port. Because if I offer something via Kafka, I have different guarantees and agreements than when I offer an anonymized version that you may use for more purposes."*

Because decoupled architectures rely so heavily on these interfaces, the technical sequencing of their implementation led some participants to the "contract-first" versus "product-first" debate. The data illustrates that traditional development often involved building the data product first and adjusting it based on the consumers' wishes. However, leveraging contracts to decouple domains encourages a contract-first approach, which most prefer. By defining the requirements, schema, and quality guarantees upfront, producers and consumers establish a clear agreement that dictates and streamlines the engineering of the data product. A critical perspective emerged that this upfront alignment can inadvertently constrain the data product to serving a single data consumer exclusively. Tailoring a data product to a contract creates a one-to-one solution, which diminishes the product's reusability.

> P6: *"There are two ways to define a contract, it's data-first or contract-first. There are three ways to do the contract-first. You can either as a producer say: this is what I could potentially offer. I have not built it yet, but I have it somewhere and I just put it out there in the marketplace and say I could offer that. If there's enough demand, I build it.* [...] *The other part is from the consumer side. The consumer says this is what I need. I just define what I need and then I try to find someone who can help me build it. And the third one is you already know generally where the data should be coming from, you have some consumers or producers, you put them all in a room, and you define the contract."*

> P7: *"What I recommend to people is to actually start with the data contract. When you start with the data contract, you get value very quickly out of it. You can validate, you can run your data quality checks, etc., without even having to think about it being a data product."*

> P9: *"You have companies where you actually have data contract first and data product second. That is interesting, but then in my opinion you miss that reusability a bit.* [...] *With that you actually create a one-on-one situation and that is not the point of a data product. It is good to have standardized templates in which dimensions stand that you can consider."*

Finally, to successfully transition toward a decoupled, contract-driven architecture, the experts uniformly advocated for an iterative, value-driven implementation strategy. As participants agree that a "big bang" overhaul to decentralize an entire organization simultaneously is highly ineffective and often creates organizational inertia (more on that in Section 4.1.5.1). Instead, the optimal approach, according to the experts, is to start with a minimal, functional scope that focuses on delivering immediate value. By avoiding upfront perfectionism, organizations can deploy basic contracts that are "good enough" to get started, then continuously iterate on and expand those foundations as the domain matures.

> P4: *"You cannot have a big bang is what I'm saying. So you need to have an operating model and* [...] *define the funding, governance structure, talent enablement, all of those things. And then you need to have a thin slice approach."*

> P8: *"This is all about figuring out what is the minimum, the absolute minimum set of data contract attributes that you can deliver value. Value is delivered, at least in most IT organizations and businesses, as you deliver products."*

> P10: *"We are still busy implementing our data governance program. We do that per data domain. So for a number of data domains, we have already appointed someone in the business as owner of data, but for a lot of data domains not yet.* [...] *It is all long-term stuff. So for a good implementation of data governance you need a certain critical mass performing certain actions before the organization as a whole benefits from it. And we are onboarding department by department onto the data management program.*

Taken together, the results establish that designing decoupled data architectures is the primary strategy for realizing decentralized domain autonomy and data ownership. By relying on data contracts as the foundational interfaces, organizations can safely minimize architectural coupling, dictate clear development phases, and iteratively scale their federated data ecosystem.

4.1.4.2 Structuring Human-Agent Dynamics

As organizations explore the automation of (governance) tasks through LLMs, a distinct interaction strategy emerges: architecting the collaboration between employees and AI agents. Several experts (P4, P5, P6, P8, and P11) view agents as part of or even as core participants of the system. The analysis showed that, rather than viewing AI as an autonomous replacement for humans, these participants strategically position AI to enhance human productivity, for example, by helping data stewards handle data contracts at scale. Furthermore, several experts discussed the non-deterministic nature of AI. One participant highlighted that this often overlooks the fact that people are also non-deterministic and make errors in the same processes.

> P4: *"I would use both. LLM and a data steward. Because a data steward to handle 500,000 contracts is not going to work.* [...] *So I would put it on automation plus plus, right? So automation for 80% of the time and then you handle the 20% edge cases separately.*

> P8: *"We see agents as core participants in a business process. Here is the thing everybody comes back with: the initial discussion is, they are non-deterministic, they make mistakes. I said, fair point. But I'll tell you what my clients say: 'I have 300 people doing this process right now, and guess what? Do you think those people are deterministic? No.' They make mistakes, even though they have the standard operating procedures sitting right beside them."*

Crucially, the data indicates that this human-agent collaboration relies entirely on data contracts to act as a standardized, machine-readable boundary. Because agents require structured metadata to navigate data products, they cannot operate effectively on tacit knowledge or unstructured business logic. Several experts emphasized that the formal expectations and quality guarantees codified within a data contract are precisely what can guide the AI's decisions, allowing the agent to discover data, evaluate its utility, and request access. In this context, the data contracts are not only an interface or alignment tool for humans but also become an essential interface for AI.

> P6: *"It's essential for AI. Because what does AI need? It needs data. And it needs to quickly find that data. So it does this discovery part, checks whether the data actually would make sense for it, and then it needs to request access and actually use the data. When you have this AI agent in place, the data contract is essentially the interesting information. Again, in combination with the marketplace, because you need to request access as well. But based on the contract, an AI can make a decision: is this data offering helpful for my current task? That's why this data contract is such a huge sphere, because it's so central in so many places."*

> P5: *"But if you think what agents can do for your governance and how you can accelerate creation, enforcement, filling, auditing, compliance, all these things using agents.* [...] *So unfortunately the use of AI is still on generating code from contracts or generating contract boilerplates, which is good, but it's barely scratching the surface. It is just using one percent of the potential. We are aware of what could be done. But here, in a super big company that is system-critical, in this day and age, security is super important. The same goes for system stability and stuff like that."*

> P11: *"We were trying to see if an AI is capable of, for instance, automatically identifying if a data product is GDPR-relevant, or if it needs special permissions. And basically, it works. The idea was that we could make automatic approvals, for instance."*

To safely operationalize this collaboration and mitigate the trust and hallucination barriers in AI (detailed further in Section 4.1.5.2), participants outlined a layered strategy. The data illustrates that deterministic rules, such as schema registries and hard constraints explicitly defined in the data contract, must form the primary enforcement layer. The agent then operates on top of this foundation, utilizing the structured metadata. Throughout this process, organizations must strictly maintain a human-in-the-loop requirement to manage edge cases, verify the agent's interpretation, and ensure compliance with strict legal mandates. Just as data contracts establish trust between human teams, parallel frameworks must be built to do the same between agents and people.

> P4: *"You have an Enforcement Layer, hard checks where you have constraints, schema registry, policy tags, CI/CD gates, whatever. But these are all non-LLM.* [...] *Because then you have an LLM Agent Layer. Where you have one or more agents, that take that*

> *contract and convert it into, or translate it into executable checks, talk about impact, talk about compatibility, where can I go for the human. So one very important thing is of course always allow a human-in-the-loop."*
>
> P8: *"You still need to have a trust framework just like you have to trust data; there is a different trust framework for agents."*

Structuring human-agent dynamics is the next step for scaling governance, one that is fundamentally dependent on the clarity and centrality of data contracts. By using machine-readable contracts, layering deterministic checks, and allowing human oversight for complex cases, organizations can leverage agents while remaining compliant with legal mandates and trust requirements.

4.1.4.3 Standardizing Contract Specification

How a data contract is specified and defined remains an ongoing debate among experts. Every participant discussed either how it is currently specified or how they would envision the ideal data contract. Many of the participants strongly advocated for an "Everything as Code" approach, standardizing contracts in structured, machine-readable formats such as YAML or JSON and storing them in version-controlled systems. The data shows that this structured approach is favored because it establishes deterministic rules, removes ambiguity, and enables the automated validation required to scale decentralized systems. Version control is treated as essential rather than incidental: because both the underlying data and the business requirements it serves change continuously, participants framed versioning as the mechanism that keeps a contract aligned with its data product over time.

> P2: *"We have a number of principles. One of those is 'Everything as Code'. So we want to do as much as possible with version control, capture as much as possible in code.* [...] *You do want to open up the world to non-technical contributors. Everything must be versioned anyway, so everything must be in Git. But it must be accessible for non-technical people who don't know their way around Git or don't understand what they are looking at."*
>
> P6: *"I think the key part is that because it's machine readable, we can validate it and we can use it for automation. So I think that's crucial to also structure it, to make it comparable and standard compliant. And basically what we've seen, which is also an interesting fact, is that a lot of companies build their own data contract format and all of them basically choose a structured format. Maybe Excel even, or JSON. But all choose the structured format because then you can work with it."*
>
> P10: *"That is all kept in Git. They are all YAML files. So every data contract is a YAML file. I just type the YAML file with a description of my information. And the rest goes automatically."*

A critical perspective also emerged from the data, highlighting that code-based standardizations can create barriers for non-technical users. While engineers prefer working with command-line interfaces and Git repositories, several experts noted that enforcing this technical medium isolates the business domains that are ultimately responsible for the data. A subset of participants advocates for lower-barrier formats, such as Excel or Word documents or conversational natural-language interfaces, to ensure that business stakeholders can actively participate in drafting these contracts.

> P5: *"It might be, I am just saying something maybe a little bit sounding like blasphemy, but it could be a simple Excel file between us that we agree that we always keep it like this, right? You are agreeing on some basics. It is like going to a notary and signing something. Does it matter if it is electronic or on paper? No. But you should have some sort of contractual agreement and signing process acknowledging that."*

> P8: *"You're going to smile at this or laugh perhaps... they are built in Excel. First, I had the same reaction, but here is why: it is not a data exercise. It is a "what are my service expectations" exercise. "What are my quality expectations?" And then translating that. If you can capture those things, that is a Word document."*

However, some participants acknowledged the long-term limitations of this approach. While unstructured formats provide an accessible starting point, they ultimately hinder scalability because they cannot be reliably automated.

> P11: *"We started with Confluence. Doing data contracts there was a pain. It was a good start, better than nothing, but it turns out you cannot really automate things on top of this kind of software because the users make things a little different. Putting it in Word documents or Confluence documents was a dead end."*

To combat the need for early business accessibility and eventual technical enforcement, the findings indicate a shift towards semantics over syntax. Participants emphasized that the specific file format is less important than the underlying semantic agreement and the capture of business concepts. From the data, we can conclude that a data contract is, in essence, a formalized promise between a data producer and consumer; therefore, defining the business meaning, quality targets, and decision boundaries is more critical than whether the contract is a YAML, JSON, or Excel file.

> P5: *"The semantic and process of creating data contracts is way more important than JSON, BSON, YAML, Kafka topics, what. Because you can take the same semantic and agree on that and say with one button click you can transform it to a JSON or to a YAML or to whatever you want. All of them are the same things with different manifestations."*

> P8: *"Data contracts, if you want to do them well, you have to capture the concepts, but you have to capture the policies and the decision boundaries that the actual business runs in."*

An essential component of formalizing the semantic agreements in a data contract is specifying purpose-binding and consumer obligations. The data illustrate that a contract is not only about defining the data assets a producer provides but also about documenting exactly why the consumer is requesting the data and what restrictions apply to its subsequent use. This trust relationship relies on explicitly registering use cases, ensuring that data is consumed responsibly and in compliance, and generating real organizational value.

> P6: *"You even have expectations on the consumer. There are limitations. You are not allowed to do just anything with the data, it's restricted. Maybe you are not allowed to use it for marketing purposes. So when the consumer arrives, they basically have to say: I also fulfill my part. Not only the producer of the data, the data owner, has to fulfill their part, but it's also a fulfillment for the consumers."*

> P9: *"A data contract is about agreements or, if you want to flip it, promises. And then I see 2 types of promises: You have promises that I make as a provider to you as a consumer, and you have promises that you as a consumer make to me as a provider. I promise the quality is this high and that you may call me, those are typical SLA things. You promise that you use the data for this and not for that, that you do not share the data.* [...] *A promise can thus also be that I describe a good use case and that you can use that to prove that your data product generates value. I would like to capture those kinds of agreements in a data contract."*

The results establish that standardizing specifications is an interaction strategy requiring organizations to balance machine-readable technicalities with human-readable semantic clarity. By defining structured, purpose-bound contracts that prioritize business meaning

over syntactical preferences, organizations can successfully bridge the gap between technical enforcement and ownership. This specification not only aligns decentralized teams but also establishes the essential, machine-interpretable foundation required to scale discovery and governance workflows.

4.1.4.4 Facilitating Interpersonal Alignment

Teams within decentralized architectures often operate with localized tacit knowledge that is not easily understood by downstream consumers. To mitigate this cross-domain friction, a data contract is also used to initiate the conversation between the data producer and the consumer. The analysis indicates that the contract serves as a discussion template that translates local contexts into shared understanding about the data.

> P7: *"For me, I see the data contract as a framework for a discussion between engineers and business. There are a lot of sections in the data contract that you can follow as a discussion between the two parties.* [...] *As you follow the contract as a template, you have to answer these questions, but you're not forced. We're not forcing you to answer all the questions because the idea is also to be able to have an iterative process around the contract"*

> P4: *"What I have seen in my experience, how can we best do this dance... so it is called a Tango. It takes a producer and a consumer. So, okay, a producer is producing data, consumer wants this, let's bring them together and make them talk to one another."*

Beyond exchanging technical definitions, the data demonstrate that using the contract as a bridge enables teams to establish mutual, human-readable promises. This was also briefly mentioned in the previous section, but since that section is more about semantics, this is about alignment between domains and whether the data is really necessary for the use case. This bilateral agreement prevents unspoken assumptions and aligns the views of both parties.

> P3: *"What I ultimately want is that you put a request with a certain business justification, so a certain need, why do you want access to that data, and that you go to an owner with that. And that they press 'approve' and then you get an email saying: hey, you have access to this for six months."*

> P10: *"We also bring them* [data producers] *together with their consumers, with the people who use the data.* [...] *That the consumers also often have assumptions and presuppositions or don't quite understand what they are looking at."*

A critical perspective emerged during the discussion of automating this alignment process. Because data is increasingly managed by business stakeholders rather than IT professionals, the contract language should remain accessible to non-technical stakeholders. This participant indicated that while technical execution can be automated, the initial negotiation of objectives, agreements, and usage boundaries inherently requires human conversation and cannot be done by AI at this point.

> P9: *"If we talk about SLOs* [service-level objectives] *and quality guarantees, then those must be defined by humans for the time being. You can say AI can generate it. But the first time you do it, you have to have a conversation about that, because you can only have the quality of data up to a certain level, depending on what your source is. The enforcement and measuring of that may all be automatic, but that is not the essence of a data contract for me.* [...] *A data contract is that collection of promises between the consumer and the provider, which is initially something between humans or between AIs, but not something that is fully automatable for the time being."*

The results establish that data contracts can serve as objects to draw clear boundaries. As the data contract becomes more central within an organization, it enables isolated teams to initiate structured conversations that facilitate interpersonal alignment, translating tacit business knowledge into explicit agreements that, in turn, increase trust between domains.

4.1.5 Intervening Conditions

Intervening conditions represent the factors that constrain the action and interaction strategies deployed to manage the core phenomenon. While the centrality of data contracts provides a foundation for distributed architectures, the data reveal that organizations still face practical barriers to operationalizing and automating these architectures. The transition from centralized to decentralized does not occur on its own; rather, it is heavily influenced by internal organizational cultures and external technology. The data identified three primary intervening conditions that currently disrupt this transition. First, the human and cultural aspects, where passive resistance manifests as organizational inertia. Secondly, the current limitations of AI models, and finally, the immaturity of the decentralized data architectures industry. Together, these factors explain why moving from data contracts to automated, executable enforcement remains a complex engineering and change-management challenge rather than a "simple" architectural shift.

4.1.5.1 Organizational Inertia & Constraints

All participants, except P6, discussed how organizational culture and human tendencies have a severe impact on the integration of new concepts, such as data contracts. P11 summarized this core friction perfectly, noting: *"If you want to move forward, you face organizational inertia. There is an inertia to stay with existing structures, processes, and cultures, and this is the most complicated part."*

A significant barrier to adopting federated governance is the historical legacy of centralized control. Several participants observed that organizational leaders and data professionals are familiar with a "police-first" culture in which a central authority dictates governance. Transitioning away from this model requires a cultural shift that is frequently met with resistance; business units are often hesitant to embrace new frameworks, while IT departments struggle to provide control. Furthermore, one participant noted that this resistance is sometimes exacerbated by leadership volatility, where employees deliberately ignore new initiatives, anticipating that the executives supporting them will soon depart.

> P5: *"Most of people who are in charge and decision makers in this time, are people who started their careers 20 years ago, 15 years ago, 10 years ago, and those people should be very self conscious that they are grown in a very police first culture, that you had a centralized team who were policing everything. So I should be very self aware to stop it and say that this is a new time, and I am going to let the new organizations to be self policing."*

> P7: *"this is this CEO that is coming here and he wants to implement this thing. If I just pretend I don't listen to him, we'll stay as it is, it's already working. So why should I change something? There's really this passive resistance in a lot of companies that just say, yeah, it's not mainstream, you're taking a little bit of a risk here. And you know, the life expectation of a CEO in a company is between 18 months and 24 months. So yeah, I'm just going to wait it out."*

Even when organizations attempt to implement data contracts to mitigate tension between domains, the data reveal an incentive gap between governance goals and domain teams' operational priorities. Participants highlighted that establishing data contracts is often perceived as mandatory and an unrewarding administrative chore. Data producers

often struggle to see immediate value in taking this responsibility of formally owning and documenting their data products. The findings suggest that overcoming this inertia requires governance teams to motivate employees and answer the "What's in it for me?" question.

> P4: *"I use something called WIIFM. That's "What's In It For Me" analysis. So I build something and tell people, okay guys, we have a nice thing here, this is what we going to do. And this is what we're going to build. But what's in it for me? So then you explain to them: Ah, this is what it can help you with. I'm going to ask, I'm going to enforce this, but this is what you'll get if you follow this."*

> P9: *"The Data Delivery Agreements have a standard template that you can fill in. It is often seen as a mandatory chore.* [...] *What is really difficult in many organizations, is to make people enthusiastic about this. I have a passion for data products and data mesh, I think it's cool. But people often don't want to understand it.* [...] *Nobody lies awake at night because of data quality."*

> P10: *"We have an enormous 'grocer's mentality'. So if you want to sell me something today, I need to know what it will yield me tomorrow. So you also very often get the question: hey, you come with a whole package of tasks, but what's in it for me? And that is difficult to make tangible on the short term with data governance."*

> P11: *"They say, 'I have all the costs. I need to implement data quality. I need to maintain this data contract. Now I need to upload it into the data catalog. What is my benefit?' The benefit is for everyone else in the organization. It will be valuable to someone else, and it creates better data quality. But there is no immediate return on investment for my particular team. It's a big problem. The entire organization benefits, but you have budget constraints at the same time. In this situation, you really need top management backing."*

The findings establish that the practical formalization of data contracts is significantly constrained by the organization's reliance on tacit knowledge. Data definitions, business rules, and downstream dependencies are rarely documented; instead, this knowledge is contained in the minds of individual employees. Translating this understanding into specifications introduces friction. One participant illustrated that human interpretations of data can vary over time, making it difficult to establish general agreement. Another expert noted that business domain specialists often lack the technical fluency to formalize their knowledge into (semi-)structured contracts. This reliance on undocumented tacit knowledge creates a huge barrier for automating data and governance workflows.

> P1: *"I think you often notice that the knowledge is in the people.* [...] *Gathering all that knowledge of all those source data in one place, that is the difficult part."*

> P5: *"You take the same missing data, ask the same person in two different days. So this Monday and 100 days later, right? Or you take the same missing data form or catalog and show it to two different experts in the logistics domain and ask them to help you filling it in. And these are the people on an organizational chart or rank very comparable, but you will see, I bet you will see, different results. So humans are much more gut-driven than what we would like to be."*

> P8: *"So here is the secret sauce when we did the data contract. It is the ability to capture knowledge about the business first, and then be able to query that knowledge in a way that you can actually understand the service expectations and the quality expectations, obviously for the quality, the fields and such that are in there, are they numeric, alphabetic, ascii whatever it may be. The substrate, the foundation that allows a data contract to work before you even think about the data contract, is you have to have some what we call "knowledge engineering."*

Finally, the analysis demonstrated that resource constraints act as a severe practical barrier to implementing data contracts. Participants emphasized that creating, negotiating, and maintaining these contracts requires time, budget, and cognitive capacity, resources that are mostly prioritized for value creation, such as feature development or AI use cases, rather than data quality. Teams simply lack the resources to adopt new workflows or learn new languages alongside their day-to-day operations.

> P2: *"Time* [talking about the biggest barrier to innovate]*. Taking the time to think this through properly, test it well together, run the proof of concept, and then productionize it."*

> P5: *"This might not be the best time for me to learn new stuff. I might not have the cognitive load of the whole thing doesn't let me to use another tool."*

> P9: *"It revolves around: can you get the executive boards to invest a lot of money, not in the AI use cases, but in data quality."*

The results establish that organizational inertia and operational constraints serve as highly disruptive intervening conditions. While decoupled architectures and standard specifications provide the foundation for federated governance, their success is dependent on the organization's ability to overcome cultural resistance, align short-term incentives, document tacit knowledge, and allocate sufficient resources. Without actively addressing these human and organizational factors, the technical implementation of data contracts risks becoming an administrative burden rather than the enabler of federated governance.

4.1.5.2 AI Limitations

A lack of trust in AI-generated outputs is a major barrier to fully automating governance, according to P1, P3, P4, P8, and P10. These participants emphasized that the non-deterministic nature of LLMs creates tension when applied to data architectures. One participant noted that verifying the accuracy of AI-generated rules is particularly difficult when dealing with legacy codebases.

> P4: *"If you go for CNL or LLM, it's of course emerging. The maturity level is emerging, it's not yet there. But of course a natural language or something native is very high, in terms of maturity. But if you go for the functional fit, it is very good. You can be quite precise from a CNL LLM side, you can be quite precise for any quality policy rules, all of that. In a natural language you have more the functional fit, if I was to call it that. But there is some hallucination risk there."*

> P10: *"AI must be correct of course. If you unleash AI on a repository you have to be certain that it doesn't spout nonsense. Is also difficult to check on a repository where a codebase sits that is 30 years old. There are also very few people who know how it worked exactly in that application. So that is perhaps the trust I think, that is currently the biggest barrier* [to automating governance]*."*

This hallucination risk is worsened by the model's knowledge boundaries. Because LLMs are limited by their training data, they lack knowledge of an organization's internal processes or public changes after the training data cutoff. To mitigate these limitations, participants highlighted the need for engineering knowledge repositories and for relying on human operators to provide accurate inputs.

> P1: *"So it is not that this LLM is not smart enough for this, but even if you run it on your own servers or something, a human still has to tell it: 'hey, this data source is going to change.' So it is about the context for the LLM because that LLM does not have in its training set that 'next week the data will be different'"*

> P8: *"The LLM only knows what it has been trained on. And there is a cutoff on the training data. Out of the box, it knows next to nothing about your company beyond what is publicly available. That knowledge repository that I keep talking about, that is the vehicle to educating, through techniques like RAG or otherwise, to actually augment the LLM with data about your company."*

Mitigating these hallucination risks and the functional boundaries of current AI models requires a structured approach to human-agent dynamics. The interview data demonstrate that a clear separation between hard, deterministic enforcement layers and AI agent layers is necessary. While agents can be useful for querying or requesting information, when they translate specifications into code, they must be constrained by safety nets and human-in-the-loop checkpoints to prevent unintended system impact.

> P4: *"Go for a Data Contract layer where you store everything as versioned YAML in Git. Of course with schemas, quality rules, privacy constraints, PII, maybe even use cases and versions and so on. That's the first layer. Then you have an Enforcement Layer, hard checks where you have constraints, schema registry, policy tags, CI/CD gates, whatever. But these are all non-LLM.* [...] *Because then you have an LLM Agent Layer. Where you have one or more agents, that take that contract and convert it into, or translate it into executable checks, talk about impact, talk about compatibility, where can I go for the human. So one very important thing is of course always allow a human-in-the-loop."*

The risks of hallucinations and the inability to natively understand dynamic contexts mean that organizations can not rely on automated agents with the current AI models. Instead, this condition necessitates architectural designs, forcing organizations to explicitly structure how people interact with agents to ensure the integrity of their data systems.

4.1.5.3 Industry Immaturity

A major obstacle to the adoption and automation of data contracts is the industry's immaturity. Highlighted by every participant except P2, current implementations of data contracts are difficult due to the immaturity and fragmentation of vendor tooling, as well as a lack of industry standards and definitions. The market offers tools capable of logging metadata or defining data quality rules, but the data reveals a lack of integrated, out-of-the-box solutions designed to enforce contract consequences. Consequently, organizations are often forced to rely on manual implementations for access control, treating current data catalogs as passive directories rather than actionable, transactional interfaces.

> P4: *"It's a really a it's a very very complicated it's like a jungle, right? You don't know. And that's a big problem today. Because I can clearly say that not many tools do that out of the box. And definitely no integration with other things."*

> P10: *"You always have to perform an extra action to gain access to that data. That is approximately what the data catalog we have now looks like. You can browse through it, you can search in it, you have all the information at your disposal that you need. But that extra step of 'order now' and 'I want to have it now', we are missing that."*

Besides the technical limitations of the tooling, the data indicated that the industry suffers from a lack of standards. Participants expressed frustration over the absence of standardized specifications, which forced early adopters to develop proprietary solutions. As large organizations use standards to scale and reduce vendor lock-in, the lack of initial industry consensus led companies to develop their own distinct, isolated formats.

> P6: *"The problem that data mesh and these data products are solving is mostly for large companies. And these really large companies, they love standards. The problem was that with the early movers, there was no standard, so they built their own."*

The fragmentation is further contextualized by the lack of industry-wide collaboration. Participant 5 drew parallels to a historical milestone of the creation of the Java Community Process, noting that the data ecosystem currently lacks a place where competitors agree on a single specification to drive this shift.

> P5: *"Sun* [the company] *deliberately made a humble choice to lower their position: 'I am just going to create a room, and this is the place that we will agree on specification.' [...] Everyone had a share in that success, and everyone understood that. That hasn't happened yet in the data contracts world."*

Recent industry developments suggest a shift towards broader standardization, such as the Open Data Contract Standard (ODCS), which aims to establish a unified framework [13]. Some participants emphasized that young standards often suffer from rapid, inconsistent versioning. This highlights that, to be practically useful, standards need to provide long-term stability rather than unified concepts.

> P8: *"Instead, where there is an immature approach, these standards that folks are creating are new. What that really means is version one maybe came out, but version two is going to be out in three months, and version three, and they are going to be different. So the ability to link into a standard is great when it's not just coherent and consistent, but when it's stable."*

The data also revealed an ongoing discussion about whether a data product should be defined strictly by its input and output ports. Additionally, there is a disconnect regarding granularity, arguing that a single product requires multiple contracts to handle different consumer interfaces.

> P9: *"I find that there is a shortage of good definitions of what a* [data] *product is holistically. There are few people, there are some, but who think in data products with input and output ports."*

> P3: *"In my mental model, a data product can have multiple data contracts. I see online that data products actually always have one data contract. I find that questionable.* [...] *I really think that if you have a customer, that you as a team should be responsible for the "customer" data product. And you have different interfaces for that."*

The industry's immaturity severely constrains the adoption and implementation of decentralized governance. The combination of fragmented vendor capabilities and not-yet-stable standards prevents operationalization. This immaturity dictates that standardizing contract specifications and designing distributed architectures currently remain complex engineering challenges rather than a scalable, plug-and-play reality.

4.1.6 Consequences

Consequences in the axial coding paradigm represent the outcomes of action and interaction strategies. The findings from the data suggest that navigating the federated governance landscape produces organizational outcomes as the core phenomenon, centrality of data contracts, which yield three consequences for the distributed data architecture. The first is that it enables the operationalization of rules, leading to governance automation. Secondly, the data contracts allow for structured metadata necessary to facilitate complex data discovery and workflows. And finally, it establishes codified trust, replacing social trust with technical guarantees. The following subsections detail how these three consequences transform the operations of a data ecosystem.

4.1.6.1 Governance Automation

The shift from static, document-based governance toward operationalizing contracts as executable code is discussed by every participant. They highlighted that treating the contract as a machine-readable artifact allows organizations to embed governance into the development lifecycle. This enables CI/CD pipelines to automatically validate data against the contract, simplifying DataOps and preventing non-compliant changes from reaching production.

> P2: *"I would very much like, as a next step, to be able to integrate data contracts into pipelines where developers can already see before they push code whether they are breaking data contracts or not."*

> P6: *"The other aspect of the contract is that it's machine-readable. And that's why you can verify whether the data matches the contract. That helps establish trust, because you can really rely on what the owner promises.* [...] *I think the key part is that because it's machine readable, we can validate it and we can use it for automation. So I think that's crucial to also structure it, to make it comparable and standard compliant."*

> P7: *"From a pragmatic perspective within the company, what I'm seeing is that it simplifies DataOps. People are looking at the contract and they can directly control things from the contracts, like data quality and validation of the data."*

Building upon this executable foundation, the data illustrates that effective governance automation requires engineering of machine-readable business rules and data quality checks. The findings suggest that simply defining a schema is insufficient; organizations should clearly define SLAs, SLOs, and semantic ontologies so that platforms can automatically measure compliance. By codifying these targets, the underlying data platforms can monitor thresholds, identify gaps, and enforce policies without requiring manual interventions.

> P8: *"Enforcing governance, there are two steps to it. Actually, there are three or four steps. The very first step is to say: what does "good" look like? A rubric. The second step is: now that I know what good looks like, how do I measure it? And then once I have measured it, the next step is: how do I identify the gaps, where are the non-conformances? And the fourth step is: how do I teach, or improve, or optimize the process after the fact? You have to have all four of those steps in place if you want to actually enforce governance.* [...] *So if you want to enforce governance, you have to actually move governance in a federated fashion to where the data products or the data lives. As close as possible to the data. And once you can do that, then you can apply the thing that I mentioned: the rubrics, the measurement, the conformance, and then the optimization or teaching. Only then can you have the appropriate knowledge to do that."*

> P9: *"Quality checks, the SLA and SLO dimensions, you can automate. We have dashboards for that.* [...] *Then you also have the legal aspect and the lineage are also super important, then you go more to the technical side, you have business lineage, your logical lineage, your technical lineage. But lineage I would automate completely from the technical perspective by the way."*

Participants also emphasized that combining machine-readable contracts with LLMs introduces a new type of automation: AI-driven automation. Several experts described using LLMs as copilots or agents that can parse metadata repositories, translate natural-language requirements into technical YAML or JSON formats, and facilitate automated access requests. However, as noted in Section 4.1.3 regarding legal mandates, participants explicitly mentioned that this automation is not absolute; a human-in-the-loop remains structurally necessary to approve purpose-binding requests and comply with regulations.

> P8: *"We use LLMs to help us out do this and it is a remarkable advancement.* [...] *Almost all is automated. And anybody who is not automating this is missing a huge opportunity.* [...] *Now I can ask the LLM: what are the service level expectations? Tell me about the data attributes that you heard. Tell me where the conflicts may be where I need to actually ask some more questions. Tell me the different senses of data."*

> P4: *"For me the data contract should be treated as code. Like I said then the LLM can be that assistant layer or policy layer that you interpret, enforce, whatever you say. Then you have CI/CD, then you have runtime access, you have all of that.* [...] *For me an LLM is like a copilot for the life cycle of the contract, for compliance, for evidence, for whatever reason. It is a copilot to help me, go through this, the human and CI/CD gates are the breaks."*

Governance automation is not only about adopting new tooling but also a structural consequence of defining data contracts as machine-readable code. By transforming agreements into executable code, engineering quality checks, and integrating AI capabilities next to employees, organizations can effectively scale their federated governance models without sacrificing control or agility.

4.1.6.2 Data Discovery & Workflows

The transition to decentralized data architectures necessitates a centralized discovery mechanism, which the data indicates is most often realized through a data marketplace. Several participants explicitly linked the data marketplace back to the core phenomenon, noting that it is powered by aggregating data contracts into a shared catalog. By using contracts as the underlying metadata, organizations can unlock data product discovery across the organization. However, participants discussed that for domain teams to share and consume data, the marketplace must prioritize the Developer Experience (DX). If a platform is too complex or reliant on technical syntax, users will bypass the governed system, creating shadow behaviors. By optimizing the platform, organizations can reduce the friction associated with finding, evaluating, and ultimately consuming data.

> P2: *" I think you should have one data platform, but that people can work with it decentrally. If you do that well, then my motto is: "Life has to be better on the platform than off the platform." Why do you buy on Amazon? Why do you buy on eBay or Marktplaats? Why don't you go to the local supermarket and see what note is hanging there saying "Hi, I'm selling my LG TV"? You don't do that anymore, because the friction to use a marketplace, where you can search, find, get in touch, order, and get it delivered home, you want to apply that analogy to your data platform too."*

> P3: *"This year, based on the data contracts we have, we are actually going to say: okay, we are going to unlock some of those data products in a Data Marketplace. So that you can actually say: hey, I want access"*

> P5: *"Most of the data platform tools are really terrible. I can tell you from first hand experience that people who are designing it, they have no idea about DX, or developer experience.* [...] *People should feel powerful. I am using this tool, I am on top of it. If you feel that I need to jump through 80 hoops to do the things that I used to do much easier, I will find a way not to do it"*

> P11: *"We could strictly enforce that if you are on our platform, you are obliged to use it, that will be the enforcement policy. But if we do, a business department might just go to another platform and build a shadow IT installation somewhere out of our reach. That happens. But the new platform will be fancy enough that they won't want to do things anywhere else anymore."*

Beyond simply finding data, the participants discussed how these marketplaces operationalize governance through integrated use-case registries and access workflows. The interview data indicate that consumers cannot simply extract data; they must explicitly justify their requirements through purpose-bound requests. These requests are routed to the data owner for approval, creating a continuous governance cycle from producer to consumer. One participant noted that this strict registry mechanism also provides the telemetry needed to identify unused tables and proactively revoke access rights if query logs do not match approved use-case scopes.

> P3: *"What I ultimately want is that you put a request with a certain business justification, so a certain need, why do you want access to that data, and that you go to an owner with that. And that they press 'approve' and then you get an email saying: hey, you have access to this for six months. That used to all go to the data owner, then the head of risk. So here the approval will be given by the data owner. That is recorded by Data Governance, so the data governance practice."*

> P10: *"If there is no use case containing that table, then we can assume that that table is not being used anywhere. But we can then subsequently see, the moment people have requested access to that table for a use case, but if we don't see in the logs that that table is being called for that use case, then we can also see: hey, you probably requested too much access in your use case. You don't need that. So then we can downscale that again."*

> P9: *"You register a use case, you say that you need certain Golden Data Sets, and you indicate down to row or column level what you want to have. Then the data stewards of the corresponding data domains get a notification and they have to approve that or not."*

Implementing data contracts within the organization enables agent-first architectures. Participants argued that the machine-readable contracts can transform data marketplaces from a (mostly) human interface to an application navigable by AI agents. These agents would be able to autonomously discover relevant data products, evaluate quality guarantees within the contract, and independently request access to execute specific analytical workflows. This shifts the organizational metadata layer into an active intelligence system, reducing the reliance on manual data discovery.

> P6: *"It's always discovery, then making the decision: do we need this data product or this contract, or should we look for something else, or should we combine two? And then requesting access and actually using it. I see this more and more as this thin layer on top of the data platform. It doesn't have the actual data, just the metadata, and people sit on top, but also AI. AI is becoming the primary user at some point."*

> P8: *"If you want the agent to do things well, the agent has to have the knowledge in a queryable format so it can actually fulfill that context. So as a result of the agent work that we are doing, we backed into the knowledge engineering, which allowed us to actually look at data contracts. So the key to success is a queryable knowledge repository which allows you to pivot very quickly, answer questions around what are the service expectations for a data contract? What are the quality expectations? What is the lineage requirement? Where does the data come from? All those questions are locked in the enterprise somewhere, whether it is a standard operating procedure, tacit knowledge, metadata."*

> P9: *"I prefer data contracts to be AI-readable, because I want the AI to be able to request independent access to my data product later, by saying: I need this information for this analysis. The AI is then the consumer."*

Data discovery and workflows are being transformed by the application of data contracts. By leveraging centralized contract repositories to build intuitive marketplaces and integrated access registries, organizations can prevent shadow behaviors and enforce governance at the point of consumption. Furthermore, building these workflows on machine-readable metadata introduces the possibilities for agent-driven architectures, transitioning data platforms from static catalogs to AI-navigable applications.

4.1.6.3 Codified Trust

From the data, we can see that implementing data contracts shifts trust between domains from social reliance to technical enforcement. When expectations regarding data quality and delivery are explicitly defined, organizations can transition to a model in which trust is placed in automated tooling and the contract itself, rather than in the individual data producer.

> P5: *"If the tool says that the data is clean, the data has this level of quality, based on this universal quality framework that we agreed on, then we will both agree on that. So we are not trusting each other or each departments, we are trusting a tool. And that tool can be manifested as data contracts."*

> P6: *"The other aspect of the contract is that it's machine-readable. And that's why you can verify whether the data matches the contract. That helps establish trust, because you can really rely on what the owner promises."*

Beyond simple data quality validations, the findings indicate that codified trust relies heavily on transparency, consistency, and traceability. Multiple participants reported that trust in a decentralized architecture requires a clear understanding of data lineage and provenance. The data contract serves as the mechanism to document this journey, ensuring that downstream consumers have a record of where the data originated and how it is transformed. By codifying these elements in the contracts, organizations mitigate the risks of cross-domain friction, increase the trust boundaries, and establish a reliable foundation for data discovery.

> P8: *"Trust in data means different things to different people. If I were to look at it from a security perspective, trust means that that data is protected and that I know my customer information, or my information, is not going to be sent where it shouldn't be. That implies a whole bunch of technical capabilities and privacy capabilities.* [...] *Trust also means we understand the provenance and lineage of the data as it got transformed across the organization, and that we know there are checks and balances to ensure that happens correctly."*

> P7: *"I think this is what I try to do when it comes to building trust with a data set, a data product, a data contract, or whatever. It's not because of the value you put in it, but it's really about these three things. One is transparency, the second is expertise, and the third one is consistency. The first is transparency, it's a positive relationship. You're trying to have a positive relationship. It means: "I have a problem with my data, sorry, but here is the data anyway". Just by admitting that, you show expertise, you show a positive intent with your customers. And if you do that in a consistent way, like you measure on a regular basis, then you build trust there.* [...] *From a strategic perspective, it's about trust. All these three values of trust, positive intent, expertise, and consistency, are defined in the contract, and you can evolve them."*

Finally, the interview data reveal that codifying trust comes through mutual expectations. Participants drew parallels between the formal SLAs historically used for software applications and the requirements for data. It illustrates that a contract formally binds producers to specific delivery and quality objectives while binding consumers to specific usage constraints.

By translating these expectations into formal agreements, organizations can hold domains explicitly accountable when something goes wrong, effectively taking ownership in the federated landscape.

> P4: *"Say this is the data, this will be refreshed once a week, it is available 99.99%. When this contract is in breach, contact this person. There will be an owner of the data product and whatnot. That's how you do it."*

> P10: *"A data contract is also something where you have a certain mutual expectation from people... so you can subsequently hold them to it. It is very normal for us to agree on SLAs for applications. If an application doesn't work, then we have L1, L2, L3 support 24/7, you name it. But we don't have that so explicitly on data."*

Codified trust is a critical consequence of implementing data contracts within a distributed architecture. By converting interpersonal promises into machine-readable guarantees, mandating transparency through lineage, and enforcing mutual SLOs and SLAs, organizations mitigate the uncertainty of ownership in decentralized data architectures. This codified trust acts as the foundation for the other consequences, directly enabling automated governance and enhancing data discovery and workflows.

4.2 Exploratory Interview Findings

The transition toward decentralized data architectures is not only a technological shift but also an organizational change that relies heavily on data contracts. Organizations are initially driven towards these contracts by a struggle between centralized governance, which can create an operational bottleneck, and domain-driven agility, which can lead to fragmentation and siloes. By establishing clear boundaries and explicit ownership, data contracts resolve this tension, allowing organizations to scale without centralizing their data teams.

However, this transition does not occur alone. The implementation of data contracts is highly bound by strict legal and geopolitical mandates, such as the GDPR and internal data requirements. These regulations necessitate strict human accountability and purpose for access, limiting the extent to which data governance workflows can be fully handed over to automated systems.

As practitioners attempt to operationalize this architecture, they encounter significant practical barriers. The primary challenge is organizational inertia, where companies struggle to move away from legacy governance cultures and must rely on undocumented, tacit knowledge. Furthermore, attempts to leverage AI are hindered by a widespread lack of trust, driven by hallucination risks and by LLMs' inability to understand dynamic and internal business contexts. This is made worse by the industry's continued maturation, with tooling highly fragmented and no proven standard yet[2].

To overcome these constraints, organizations employ various operational approaches. They use a 'contract-first' methodology, which ensures domains can function independently while maintaining a reliable interface. Practitioners use hard-coded, deterministic enforcement checks to safely scale automation when employing human-agent dynamics, such as positioning AI as an assistive layer within the architecture. Crucially, when formalizing the data contracts, experts emphasize standardizing semantics over syntax. By focusing on capturing business meaning and obligations rather than dictating a specific file format, the drafting process inherently facilitates interpersonal alignment and forces conversations between data producers and consumers.

Ultimately, when these strategies successfully place data contracts at the center of the organization, they have three transformative consequences. First, they enable governance automation. While data contracts are often initially drafted in accessible, unstructured

[2] Bitol's ODCS represents a promising framework [13]

formats to lower the barrier for non-technical stakeholders, these static documents can hinder scalability. The true consequence of automation emerges when organizations transition toward operationalizing these agreements as machine-readable code. By treating contracts as executable artifacts, teams can automate quality checks and compliance validations. Second, this structured metadata enables the data marketplace to leverage these documents for enterprise-wide data discovery and purpose-driven access workflows that AI-agents can easily navigate. Most importantly, data contracts establish trust across the federated landscape, successfully replacing human assumptions with transparent, technically enforced rules, obligations, SLAs, and SLOs.

4.3 Data Contract Definition

4.3.1 Practitioner's Definition

Data contracts serve as the socio-technical bridge that resolves the tension between centralized governance and domain autonomy. In these federated environments, independent business units act as autonomous data producers and risk downstream breakage if changes are made to systems without proper oversight. Data contracts provide the mechanism to safely execute this decentralized model by ensuring reliable data exchange, mitigating cross-domain friction, and allowing autonomous domains to operate with decoupled, independent architectures.

At their core, data contracts are formalized promises and agreements between a data producer and a data consumer. They function as universal, official interfaces that define the boundaries, behavior, and structural expectations of a dataset. Rather than existing only as a technical artifact, a data contract bridges technical execution and business reality, serving as a framework for discussion that facilitates interpersonal alignment, trust, and mutual understanding across disconnected organizational domains.

Based on the interviews, a data contract consists of several elements:

- **Data Interface**: The specific technical access point or medium through which the data is offered to the consumer, such as an API or SQL database. It also includes the specific version of the dataset being shared.
- **Business Description and Metadata**: The documentation describing the data product, including who owns the data, the specific data attributes, and the technical and logical lineage.
- **Quality Guarantees**: Explicit targets and rules, including SLOs, that establish the acceptable quality, condition, and format of the data being provided.
- **Service Level Agreements**: Operational commitments regarding data delivery, uptime, refresh rates, and designated support contacts to hold domains accountable.
- **Consumer Obligations**: Explicit rules binding the consumer to responsible usage, particularly regarding purpose-binding, which defines acceptable business justifications, registered use cases, price of usage, and restrictions on how the data may be applied.
- **Security and Privacy Classifications**: The required legal and regulatory parameters, such as noting if the data contains Personally Identifiable Information (PII) or confidential material that demands specific permissions.

Within the data architecture, data contracts serve as the definitive boundary between data products. A single data product may possess multiple distinct contracts corresponding to its own output ports and consumer interfaces. Throughout the data creation lifecycle, these contracts act as the machine-readable foundation for governance automation. Because the contracts operate in highly dynamic environments, it is essential that these contracts

are version-controlled, often managed as code within repositories like Git, and explicitly define the version of the data product itself. By enforcing strict versioning, organizations can effectively combat breaking changes, ensuring downstream dependencies remain protected when source systems are modified. This lifecycle allows engineering teams to embed data quality checks and compliance validations to prevent non-compliant changes from reaching production. Furthermore, data contracts enhance the usability of data marketplaces by providing the essential metadata layer that facilitates data discovery and autonomous navigation for AI agents.

From a non-technical perspective, a data contract serves as a formal governance instrument for navigating strict legal and regulatory mandates. The contract captures the expectations, such as privacy rules, data usage limitations, and accountability. By doing so, the contract establishes clear parameters for ethical data use, specifying security classifications for PII and enforcing strict policies to ensure the data is used only for authorized purposes. Finally, to ensure the regulatory rules are enforceable, the organization can choose to translate the legal concepts into machine-readable documents. This allows data contracts to serve not only as administrative documents but also as safeguards that actively protect the organization.

4.3.2 Comparison to Literature

As seen in Section 2.1.3, academic literature characterizes data contracts as formalized, written agreements that function as human- and machine-readable interfaces between data producers and data consumers. From this perspective, data contracts are the mechanisms for ensuring reliable, high-quality data exchange between data products. The core components identified by the literature are unique identification, access mechanisms, ownership details, schema and semantics, data quality standards, and SLAs. Architecturally, the data contracts are explicitly linked to data products, either as an integrated API that enforces machine-readable constraints or as a broader data-sharing agreement.

From the interviews, we derive a similar definition of data contracts. The practitioners' view strongly aligns with the core components and functions of a data contract. Both perspectives describe data contracts as official interfaces that demarcate boundaries and define expectations for the data product. Practitioners also mention the same key elements about technical data interfaces, quality guarantees, operational SLAs, and security classifications. Furthermore, utilizing version-controlled contracts is mentioned from both directions to prevent breaking downstream dependencies. In both instances, data contracts are placed as formal governance instruments to navigate legal and regulatory mandates, enforcing privacy rules and data usage limitations.

Despite these structural and functional alignments, there is a significant deviation between the two perspectives in their prioritization of the socio-technical dimensions of the contract. While academic literature focuses more on technical formatting, architectural integration, and the automated enforcement of data contracts, practitioners frame the contracts as a socio-technical bridge designed to help address the organizational tension between centralized governance and domain autonomy. Rather than viewing data contracts as technical artifacts or machine-readable specifications, practitioners emphasize the role as a framework for discussion and interpersonal negotiation. In federated environments, the experts highlight the contract's capacity to create trust, mutual understanding, and business alignment before technical implementations. This socio-technical emphasis is clearest in the focus on consumer obligations and purpose-binding, which promote business semantics and human collaboration over technical syntax. Consequently, while the literature focuses primarily on the technical aspects of data contracts, practitioners utilize them more broadly. They leverage data contracts to resolve the complex organizational friction inherent in decentralized data management.

4.4 Problem Identification

The transition towards a distributed data architecture has shifted the responsibility of data quality and governance from centralized teams to independent domains. To manage expectations and maintain control across disconnected environments, central governance is increasingly relegated to defining regulatory guardrails, while local domains use data contracts to establish product boundaries and trust. However, the practical execution of these contracts remains fragmented. The findings from the exploratory phase identified two main barriers to automating the enforcement of data contracts and, by extension, governance: the lack of standardized contract specifications that capture both technical and legal semantics, and the severe regulatory and contextual limitations to deploying an AI agent or LLM in these environments.

4.4.1 Standardization and Translation Gap

Even though there is progress toward a universal data contract standard [13], it is not currently widely used in practice. While exploring how data contracts are currently implemented (**SRQ2**), the data revealed that this immaturity results in data contracts being drafted in varied formats, from structured formats like YAML or JSON to unstructured formats such as Word or Excel. As organizations prioritize semantics over syntax to lower the barrier for business users and capture business concepts, data contracts include not only technical SLAs but also governance and legal constraints, such as privacy regulations, consumer obligations, and purpose-binding rules (e.g., use cases). Because these legal guardrails and business rules are inside the contract, technical teams need to manually translate them into executable enforcement code and policies. This manual translation limits scalability and can create an alignment gap in which legal compliance and business intent may be lost or misinterpreted. A standardized, machine-readable intermediary that captures these governance and legal constraints alongside the technical schema would mitigate this risk, enabling automated enforcement while preserving the original intent of the contract.

4.4.2 LLM Context and Regulatory Boundary

Addressing **SRQ1**, the primary technical and organizational barriers to enforcing governance in decentralized data product architectures, multiple participants mentioned in the interviews that using AI and agentic workflows is the next step in automating the enforcement of data contracts and, more generally, governance. However, the experts caution against the direct application of AI in enforcement due to its non-deterministic behavior and strict regulatory barriers. In regulated data environments, AI cannot be trusted to make access and compliance decisions, as a human-in-the-loop is a strict requirement to ensure human accountability for the legal mandates and to verify the agent's interpretations. To operate safely in this environment, an AI agent requires a deterministic knowledge repository, which can be implemented as a layer that maps the data product mesh, including all hard constraints, lineage, and purpose-binding legal rules.

4.4.3 Problem Statement

Through the lens of DSR, these challenges should be integrated into a single core problem. Ultimately, these practical challenges highlight one central problem: the lack of a structured, intermediary semantic layer that can merge various data contract formats into a single, queryable state.

Therefore, the problem statement of this research is defined as follows:

In distributed data architectures, industry immaturity and a lack of mature standards result in data contracts being written in various formats. This creates a bottleneck, as teams must manually translate these documents into technical rules. Generative AI is seen as the future for automating this process, but it cannot be applied yet due to limitations in current models and legal requirements for human accountability. Therefore, there is a need for a framework that can parse these diverse contracts, extract their business and legal rules, and synthesize them into an intermediary, queryable knowledge base that acts as a single source of truth. Without such a reliable, machine-readable knowledge repository, AI agents lack the reliable knowledge repository necessary to safely automate data governance and discovery. Furthermore, as the number of data products grows, the absence of a unified view leaves data stewards unable to manage the increasing complexity or identify misalignments between business expectations, legal rules, and technical implementations.

4.5 Artifact Objective and Requirements

To address the practical challenges identified in the previous section. This thesis proposes the design and instantiation of an *LLM-Assisted Contract Extraction (LACE) Framework for Data Contract Enforcement*.

The primary objective of this research is to develop a KG-based intermediary framework that bridges the gap between data contracts as a concept and data contracts as machine-executable policies. The proposed artifact leverages LLMs to systematically parse data contracts, translating human semantics into deterministic, machine-readable intents. By structuring these intents into a graph topology, the artifact mitigates the barriers imposed by current industry immaturity and the non-deterministic nature of AI. Ultimately, this framework provides the deterministic, systemic foundation necessary to safely enable automated governance enforcement and agent-driven data discovery within decentralized data mesh architectures, addressing a critical need highlighted by Hechler et al., who identify automated AI governance insight components as a key research area within Data Mesh ecosystems [48].

4.5.1 Requirements

To ensure the artifact fulfills its objective and addresses the constraints identified during the exploratory interviews, the system must adhere to specific design criteria. Following the adapted requirements engineering approach defined in Section 3.3.1, these criteria are categorized into Functional Requirements (FR) and Non-Functional Requirements (NFR).

4.5.1.1 Functional Requirements

FR1: Heterogeneous data contract conversion to an unified model
Due to the lack of a mature standard for data contracts, the framework must be agnostic to input specifications. The LLM-assisted parser must be capable of extracting business intents, technical SLAs, and legal constraints from diverse document types, such as structured formats (e.g., YAML, JSON) to unstructured formats (e.g., Excel, Word documents), and mapping them into the KG.

FR2: Executable artifact generation from the unified model to enforce data contracts
To address the components required for automated governance (**SRQ4**), the resulting KG must serve as a deterministic foundation for downstream enforcement. Its topology and constraints must be structured in a machine-readable format so they can be seamlessly translated into executable policy code (e.g., Open Policy Agent/Rego rules or data quality assertions) to automate governance checks across the data mesh.

FR3: Transparency and accountability of the unified model creation and artifact generation
To facilitate human verification, the system must maintain and display a clear semantic lineage that links the generated graph attributes directly back to the original text in the source data contract.

FR4: Continuous updates to the unified model and executable artifact
The framework should support iterative updates as data contracts evolve over time. The KG should be able to dynamically update existing nodes, edges, and constraints when a modified data contract is re-parsed, without requiring a complete rebuild of the graph.

4.5.1.2 Non-Functional Requirements

NFR1: Human-in-the-Loop Quality
In response to the architectural principles required for trust (**SRQ3**) and the legal barriers of **SRQ1**, the artifact must not operate as a black box. The framework must facilitate human-in-the-loop verification. This requires having clear explainability and auditability built in, ensuring that its generated outputs can be easily interpreted and allowing data stewards or legal teams to audit, query, and approve the generated graph attributes.

NFR2: Extensible for downstream tasks beyond policy generation
To automate the governance surrounding data products (**SRQ4**), we have to look beyond policy generation; namely, the semantic structure of the KG must be optimized for AI agent traversal. The artifact must be readable by downstream agents, enabling them to traverse the graph safely to perform tasks, facilitate data discovery workflows, and answer Retrieval-Augmented Generation (RAG) queries without hallucinating.

NFR3: Semantic Accuracy & Alignment
The data extraction process must achieve semantic accuracy, ensuring that no business logic, domain-specific meaning, or legal constraints are lost, altered, or misinterpreted during KG generation. The system must maintain the meaning of all extracted relationships.

NFR4: Syntactic Accuracy & Alignment
The generated KG must strictly adhere to the defined target schema, ontology, and formatting standards, ensuring all nodes, edges, and data types are structurally valid, parseable, and perfectly aligned with the expected data model.

NFR5: Semantic & Syntactic Completeness
The extraction system must capture all relevant entities, relationships, and constraints from the source material without omission (semantic completeness), while simultaneously guaranteeing that the resulting graph contains all mandatory structural elements, fields, and logical linkages required by the schema (syntactic completeness).

4.5.2 Design Principles

Following this methodology, the Functional and Non-Functional Requirements identified were clustered into meta-requirements. These meta-requirements were then used to define the four core DPs that guide the architecture and capabilities of the LACE Framework. The resulting traceability from interview findings to requirements to design principles is summarized in Table 4.1:

DP1: Enable Unified Semantic Abstraction
Meta-Requirement: The system must be capable of processing heterogeneous input formats

without losing domain-specific meaning or structural integrity.
Derived From: FR1, NFR3, NFR4, NFR5
The framework must abstract heterogeneous, unstructured, and multi-format data contracts into a standardized, deterministic KG. This process must guarantee semantic and syntactic integrity, ensuring that no domain-specific business logic, legal constraints, or structural linkages are lost or misinterpreted during the translation from human semantics to machine-readable formats.

DP2: Facilitate Machine-Actionable Determinism
Meta-Requirement: The intermediate output must be structured in a native, machine-readable format that supports downstream autonomous execution.
Derived From: FR2, NFR2
The intermediate unified model must be structured with a graph topology that bridges the gap between static text and executable action. The artifact must support autonomous traversal by downstream AI agents for tasks like Retrieval-Augmented Generation (RAG), data discovery, and policy-as-code generation.

DP3: Ensure Traceable and Auditable Provenance
Meta-Requirement: The system must provide interpretable mechanisms to prevent "black-box" data processing.
Derived From: FR3, NFR1
To establish trust and facilitate governance, the system must avoid "black-box" operations by providing clear provenance and explainability. The framework must maintain an explicit, visualizable lineage connecting the generated graph elements directly back to the original source text, enabling human-in-the-loop verification, auditing, and approval workflows.

DP4: Support Iterative Knowledge Evolution
Meta-Requirement: The architecture must accommodate dynamic updates to reflect the continuous lifecycle of data contracts.
Derived From: FR4
The framework must treat data contracts as living documents rather than static artifacts. The underlying graph must support dynamic updates to nodes, edges, and constraints to reflect continuous changes in the source contracts over time, executing these updates locally without requiring a complete rebuild of the KG.

TAG Category	Requirements	Design Principle
Industry Immaturity & Standardizing Contract Specifications Data contracts lack standard formats, residing in heterogeneous structures (e.g., YAML, Excel), capturing both technical syntax and semantic logic.	**FR1:** Heterogeneous data contract conversion. **NFR3 & NFR4:** Semantic and Syntactic Accuracy. **NFR5:** Completeness.	**DP1:** Enable Unified Semantic Abstraction.
Governance Automation: Necessity to transition from static, document-based agreements to machine-enforceable checks and autonomous agent discovery.	**FR2:** Executable artifact generation. **NFR2:** Extensible for downstream tasks (AI navigation, RAG).	**DP2:** Facilitate Machine-Actionable Determinism.
Legal and Geopolitical Mandates & AI Limitations: Strict regulatory compliance (e.g., GDPR) demands human accountability, purpose-binding, and mitigation of AI hallucination risks.	**FR3:** Transparency and accountability of lineage. **NFR1:** Human-in-the-Loop Quality (auditability).	**DP3:** Ensure Traceable and Auditable Provenance.
Tension in Federated Governance: Cross-domain dependencies constantly evolve. Static agreements break when upstream sources change without operational synchronization.	**FR4:** Continuous updates to the unified model without requiring complete rebuilds.	**DP4:** Support Iterative Knowledge Evolution.

Table 4.1: Traceability Matrix: Mapping interview findings to artifact design principles

Chapter 5

Artifact Design and Development

Following the establishment of the DPs (design principles), this chapter details the design and instantiation of the LACE Framework. In alignment with the design and development phase of DSR, it translates the requirements into a concrete system architecture. To provide a structural blueprint of the artifact, Figure 5.1 illustrates the high-level system architecture. The framework is designed as a sequential pipeline that ingests unstructured or semi-structured data contracts, processes them through an LLM-driven semantic parser, validates the output against a strict ontology, and ultimately stores the synthesized knowledge as an executable graph. This architecture deliberately decouples the extraction logic from the underlying storage mechanism, allowing for modular updates as both LLM capabilities and semantic web standards evolve.

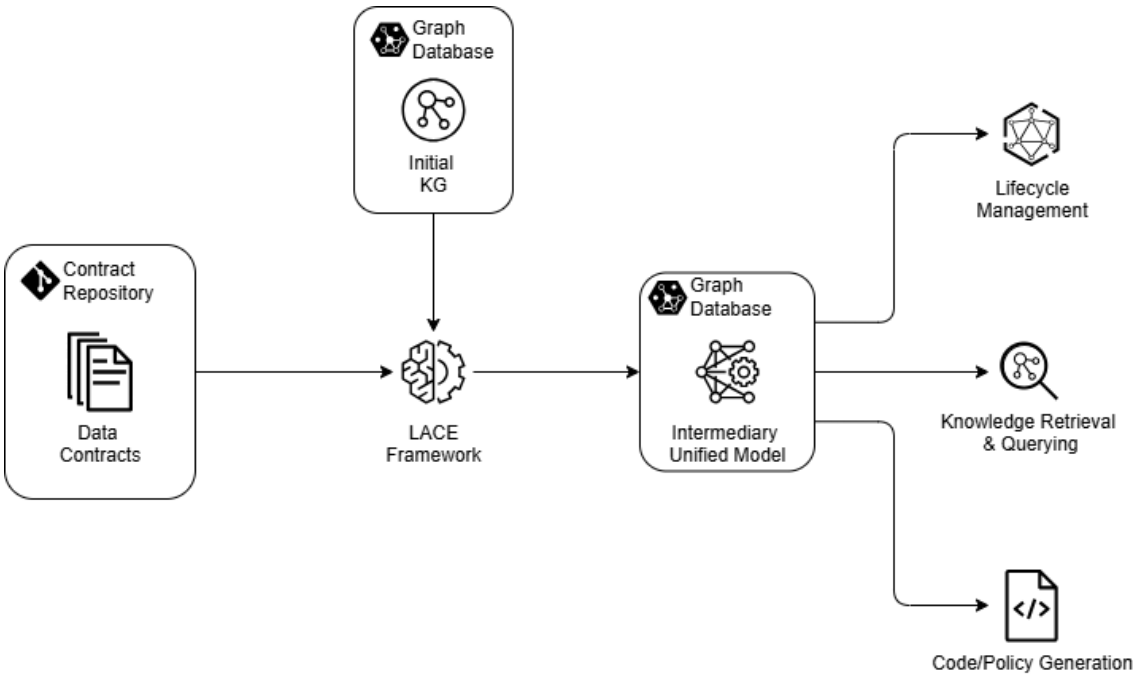

Figure 5.1: High-Level System Architecture

The remainder of this chapter is structured as follows. Section 5.1 outlines the design decisions and technical rationale that shape the artifact's components. Section 5.2 discusses the methodology and design of the data contract ontology. Section 5.3 details the system architecture, mapping the data flow from contracts to a fully validated, machine-actionable KG. Finally, Section 5.4 describes how the framework is intended to be integrated into a data mesh architecture.

5.1 Artifact Design Rationale

To ensure the artifact effectively bridges the gap between human semantics and machine-executable policies, several foundational design choices were made regarding scope, extraction mechanisms, data modeling, and validation.

5.1.1 Justification of Intermediary Knowledge Representation

The selection of an intermediary representation is a critical design decision in this research. While conventional technologies provide established paradigms for data management, they impose significant limitations for storing knowledge. Consequently, this framework adopts a KG as the core representational artifact.

Document serialization formats such as YAML or JSON could be used to represent the data contracts. While these hierarchical formats facilitate initial drafting, their lack of explicit semantic structure severely limits broad interoperability and automated governance. As the complexity of the data ecosystem grows, these formats prove unscalable for advanced querying requirements. Furthermore, relying on flat documents limits the capabilities of Retrieval-Augmented Generation (RAG) systems. In such analytical scenarios, graph-based approaches, such as GraphRAG, have been shown to outperform standard vector RAG on complex reasoning tasks, demonstrating higher factual accuracy [7].

To address the need for a more structured alternative, Relational Database Management Systems (RDBMS) are often considered. However, the relational model relies on rigid schemas and expensive table-join operations to navigate data relationships. Within a highly interconnected network of data products, this limits scalability and introduces significant performance bottlenecks. In contrast, graph databases resolve these limitations by offering flexibility and improved performance, executing queries by directly traversing connected records rather than relying on expensive joins [89]. Ultimately, because KGs inherently model the nuanced, semantic relationships among real-world entities, they are the most natural and effective artifacts for representing the complex, multi-layered dependencies of modern data products [78].

Therefore, the KG was selected as the intermediary representation. Because it addresses the semantic limitations of document formats and the computational bottlenecks of RDBMS, the KG provides a robust, scalable, and machine-actionable foundation for operationalizing data contracts within this framework.

5.1.2 Scope and Baselines Assumptions

This research is scoped to isolate its core contribution: the automated extraction and translation of data contract constraints. Therefore, the construction of a foundational, enterprise-wide KG falls outside the scope of this work. It is assumed that a structural KG, representing existing data products, assets, and dependencies, is already available. The proposed framework focuses exclusively on parsing data contracts to overlay deterministic governance, SLAs, and legal constraints onto this topology.

5.1.3 Extraction Mechanism: LLMs vs. Traditional NLP

Due to a lack of standardization and the immaturity of data contract specifications, traditional rule-based parsing approaches are insufficient to handle heterogeneous inputs. While standard Named Entity Recognition (NER) and text extraction can be performed using models such as BERT, recent literature indicates that pre-trained LLMs achieve superior performance on complex semantic parsing tasks [52]. Therefore, this framework uses LLMs as the primary mechanism for interpreting business intents and mapping them into a structured KG.

5.1.4 Knowledge Representation: RDF vs. LPGs

The Resource Description Framework (RDF) was selected as the underlying graph data model over Labeled Property Graphs (LPG). LPG formats are frequently vendor-specific, creating a risk of vendor lock-in [65]. Additionally, LPGs offer limited support for the formal knowledge representation required to conduct advanced knowledge inference [82]. In contrast, RDF is a standard developed by the World Wide Web Consortium (W3C) that uses Internationalized Resource Identifiers (IRIs) as unique identifiers. This allows the framework to facilitate data integration and interoperability across diverse systems [65].

An important distinction between the two data models is RDF's structural integration with ontologies. By providing a formal specification of concepts, an ontology defines the type hierarchies necessary to govern relationships within the graph. This formalization allows RDF graphs to execute knowledge inference and graph analytics [82]. LPGs lack this support for formal ontologies, which limits their capacity to perform automated reasoning or semantic validation [82].

Due to this integration of ontologies, RDF is often adopted in domains requiring logical reasoning, semantic search, and question answering [83]. Consequently, RDF provides a more rigorous foundation for enforcing semantic rules compared to property graphs [83].

5.1.5 Validation and Integrity: SHACL

To ensure the generated graph adheres to the defined target ontology, the Shapes Constraint Language (SHACL)[1] is applied. SHACL, a W3C standard, establishes formal schema constraints for RDF graphs using shapes. These shapes validate the structural quality of a knowledge graph and characterize its frequent patterns [40]. By implementing SHACL validation, the framework programmatically enforces NFR4: Syntactic Accuracy & Alignment (see Section 4.5.1.2). This mechanism verifies that all generated RDF triples are structurally valid, parseable, and aligned with the intended data model prior to downstream policy generation.

5.2 Ontology Engineering

Building upon the selection of RDF as the foundational data model, realizing its automated reasoning and validation capabilities requires a formally defined conceptual model: the data contract ontology. To ensure the knowledge graph accurately captures the varied nature of data contracts in a structured manner, the ontology was developed following a top-down ontology engineering approach [71]. The construction process was heavily guided by the NeOn Methodology [91], specifically adopting Scenario 2 (Reusing and Reengineering Non-Ontological Resources) and Scenario 3 (Reusing Ontological Resources). By maximizing the reuse of established W3C standards and domain-specific vocabularies, the framework promotes interoperability while extending specific classes to meet the unique requirements of decentralized data products.

5.2.1 Ontology Specification via Competency Questions

To define the scope and requirements of the knowledge graph, Competency Questions (CQs) were formulated prior to the structural modeling. According to Monfardini et al., CQs are a foundational mechanism in ontology engineering for explicitly defining the knowledge an ontology must be capable of representing and querying [71].

These questions act as the functional requirements for the semantic model, ensuring that the final graph can support tasks such as automated governance and RAG-based data

[1] https://www.w3.org/TR/shacl/

discovery. The formulated CQs dictate the classes, properties, and relationships to be extracted from the data contracts. They are organized into five categories, each targeting a distinct aspect of the data product lifecycle.

Data Products The following CQs capture the identification and business context of a data product.

1. What is the unique identifier of the data product?
2. What is the name of the data product?
3. Which business domain is associated with the data product?
4. What is the business description of the data product?
5. What is the intended use case for this data product?

Ownership These CQs identify the team and individuals accountable for a data product.

6. Which team is responsible for the data product?
7. Who is the current data product owner?
8. When did the current data product owner start?
9. What is the historical timeline of ownership?

Architecture These CQs describe the technical composition of a data product, including its datasets, dependencies, and schema.

10. Which dataset is used for the data product?
11. Which output port technologies expose the data product?
12. What are the data product's upstream dependencies?
13. What are the data product's downstream dependencies?
14. What is the dataset schema?
15. What metadata describes the schema?
16. Which fields are required in the dataset?
17. Which fields are sensitive in the dataset?

Data Contract These CQs capture the agreements that govern a data product.

18. What is the unique identifier of the data contract?
19. What are the service level agreements of the data contract?
20. What are the service level objectives of the data contract?
21. Where is the data contract located?
22. To what data products does a data contract apply?
23. Who are the parties involved in a data contract?
24. Which data products and consumers are impacted when a data contract is breached?

Governance These CQs define the rules constraining consumption of a data product.

25. Which actions are prohibited for consumers?
26. What is the expected usage of this data product?
27. Who have access to this data product?
28. Which people have approved access?
29. What obligations must a consumer agree to?

5.2.2 Reusing and Reengineering Non-Ontological Resources (NeOn Scenario 2)

Because the data mesh paradigm relies on practical, engineering-focused specifications rather than formal semantic web technologies, data contracts are frequently defined using non-ontological resources. In this framework, the ODCS was used as the primary non-ontological resource, as shown in Figure 5.2.

Following NeOn Scenario 2, the ODCS was analyzed and reengineered into a formal semantic model. The core architectural elements of the ODCS, such as schema definitions, SLAs, and stakeholder roles, were systematically mapped to target ontological concepts. This reengineering process ensures that the resulting knowledge graph remains deeply aligned with current industry standards while transforming the static ODCS fields into an operationalized KG. The concrete classes and properties that result from this mapping are presented per module in Section 5.2.5.1 through Section 5.2.5.6

5.2.3 Reuse of Existing Ontological Resources (NeOn Scenario 3)

To model the diverse components of a data contract, several established ontologies were integrated into the core schema:

- **Policies and Agreements:** The Open Digital Rights Language (ODRL)[2] ontology is used to model the policies embedded within a data contract. A data contract is represented as a policy that functions as an offer while it remains unsigned and transitions to an agreement once at least one party has formally agreed to its terms.
- **Organizational Structure and Provenance:** The W3C Organization[3] ontology is employed to define organizational units. To trace data product owners and individuals operating within these organizations, the Provenance Ontology (PROV-O)[4] is integrated. Applying this to the example in Figure 5.2, the `team` block (e.g., `my-team`) is modeled as an Organizational unit, while individual `members` like `daustin` are implemented using PROV-O to define data ownership.
- **Services and Quality:** The Data Catalog Vocabulary (DCAT)[5] is used to define data services, while the Data Quality Vocabulary (DQV)[6] provides the necessary structure to articulate specific data quality dimensions and service-level agreements.
- **Data Structure representation:** The CSV on the Web (CSVW)[7] ontology is applied to represent the tabular structure of the data. While not exclusively designed as a relational

[2] https://www.w3.org/TR/odrl-model/
[3] https://www.w3.org/TR/vocab-org/
[4] https://www.w3.org/TR/prov-o/
[5] https://www.w3.org/TR/vocab-dcat-3/
[6] https://www.w3.org/TR/vocab-dqv/
[7] https://w3c.github.io/csvw/syntax/

```yaml
version: 1.0.0
apiVersion: v3.1.0
kind: DataContract

domain: seller
dataProduct: my quantum
version: 1.1.0
status: active
id: 53581432-6c55-4ba2-a65f-72344a91553a

description:
  purpose: Views built on top of the seller tables.
  limitations: Data based on the seller perspective, no buyer information
  usage: Predict sales over time

servers:
  - server: my-postgres
    type: postgres
    host: localhost
    port: 5432
    database: pypl-edw
    schema: pp_access_views

team:
  name: my-team
  description: The team owning the data contract
  members:
    - username: daustin
      role: Owner
      description: Keeper of the grail
      dateIn: "2022-10-01"

schema:
  - id: tbl_obj
    name: tbl
    physicalName: tbl_1
    physicalType: table
    businessName: Core Payment Metrics
    description: Provides core payment metrics
    properties:
      - id: txn_ref_dt_prop
        name: transaction_reference_date
        physicalName: txn_ref_dt
        primaryKey: false
```

Figure 5.2: Open Data Contract Standard (v3.1.0) Example [13]

database ontology, it is sufficient for the structural mapping required in this work and avoids the per-schema configuration overhead of dedicated relational-to-RDF mapping languages. Future iterations could explore R2RML[8] or D2RQ[9], which are specifically optimized for mapping relational database schemas to RDF but require an explicit mapping definition per source schema that the present contract-driven approach does not.

5.2.4 Ontology Modifications and Extensions

While the framework heavily prioritizes reusing established vocabularies and adheres to the core structure of the Data Product (dprod)[10] ontology, the practical realities of data mesh implementations necessitated specific, localized design adaptations.

To fulfill operational and governance requirements without over-engineering the knowledge graph, the following architectural simplifications and extensions were implemented:

- **Data Lineage Simplification:** The standard dprod documentation prescribes a highly granular data lineage path for dependencies (i.e., `DataProduct` → `inputPort` → `isAccessServiceOf` → `isDistributionOf` → `Input Data Product`). To reduce graph traversal complexity while preserving provenance, this framework streamlines the lineage chain. Dependencies are mapped directly between interface levels using the Dublin Core `dct:source` property (i.e., `InputPort` → `dct:source` → `OutputPort`), where both ports are modeled as Data Service instances.
- **Historical Ownership:** A `ProductOwnerShip` class was designed together with a `dataProduct` object property and integrated into the graph as an extension to maintain a traceable history of data product owners of the data products.
- **Protocol Specifications:** While the Data Product ontology references interface communication protocols, a formal `dcat:Protocol` class does not exist in standard vocabularies. To address this, our framework simplifies the representation by capturing the protocol type (e.g., REST, Kafka, S3) as a standard string literal. This captures the necessary operational metadata for downstream applications without introducing unnecessary structural complexity to the core schema.
- **Data Contract Provenance:** Provenance is tracked by adding an Entity node (from the Provenance Ontology) for each parsed data contract. The file location and generation-timestamp attributes of this node link every entity created by the LLM back to its originating data contract, providing a traceable lineage from the generated triples to their source.

By combining these foundational ontology models, standard alignments, and pragmatic custom modifications, we established the final target schema utilized by the extraction framework.

5.2.5 Data Contract Ontology

This section describes and illustrates the Data Contract Ontology, which is designed to satisfy the requirements defined in Section 4.5.

The ontology was designed in a modular fashion, which provides greater maintainability, reusability, and reasoning efficiency [88]. The modular structure of the Data Contract Ontology is illustrated in Figure 5.3. The ontology integrates four modules: *Data Product*, *Ownership*, *Data Contract*, and *Dataset Schema*. Additionally, a *Provenance* sub-module integrates into each of these modules to provide a provenance lineage.

[8] https://www.w3.org/TR/r2rml/
[9] http://d2rq.org/d2rq-language
[10] https://ekgf.github.io/dprod/

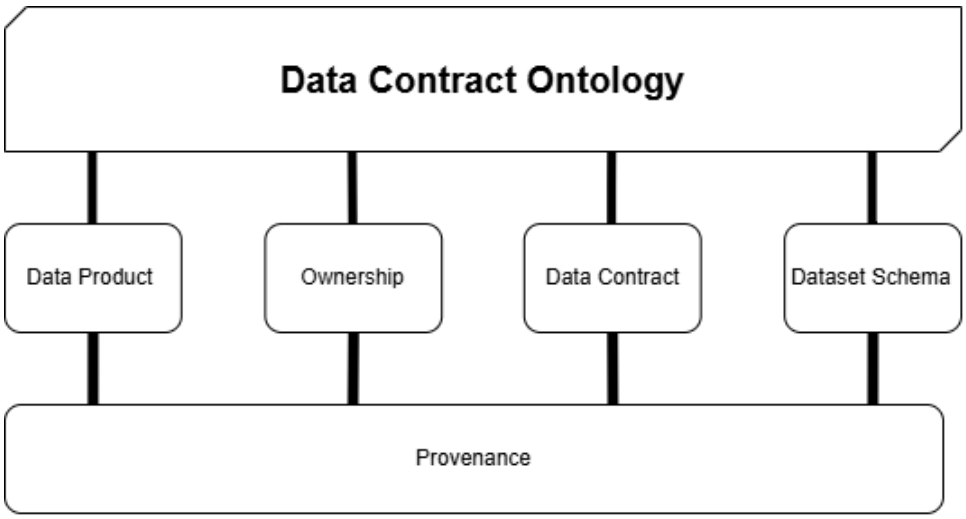

Figure 5.3: Modular structure of the Data Contract Ontology

In Section 5.2.5.1 through Section 5.2.5.5, we will describe the distinct modules through the documentation approach described by Gangemi & Presutti [35]. The concepts used to document the modules are stated in Table 5.1. Subsequently, Section 5.2.5.6 describes the integration into one coherent Data Contract Ontology.

Concept	Explanation
Name	Name of Module
Intent	Addresses the generic use-case
Competency Questions	The competency questions the module answers (Section 5.2.1)
Example Scenario	Defines natural language examples that can be modeled by the module
Diagram	An UML class diagram representing the module
Elements	Defines the classes, properties, and class hierarchies in the module
Reengineered from	Defines the reference ontology on which the module is based
Building block	Reference to the implementation of the module

Table 5.1: Documentation concepts of ontology modules [35]

5.2.5.1 Data Product Module

- **Intent**
  To formally define the boundaries of a data product, its business context, and its physical access points (ports) within the data mesh architecture.
- **Competency Questions**
  CQ1, CQ2, CQ3, CQ4, CQ5, CQ11, CQ12, CQ13
- **Elements** The classes and properties of this module, together with their hierarchy, are depicted in Figure 5.4.
- **Reengineered From**
  The core of the data product module is derived by integrating the EKGF Data Product Vocabulary (`dprod`) and the W3C Data Catalog Vocabulary (DCAT). The data product itself

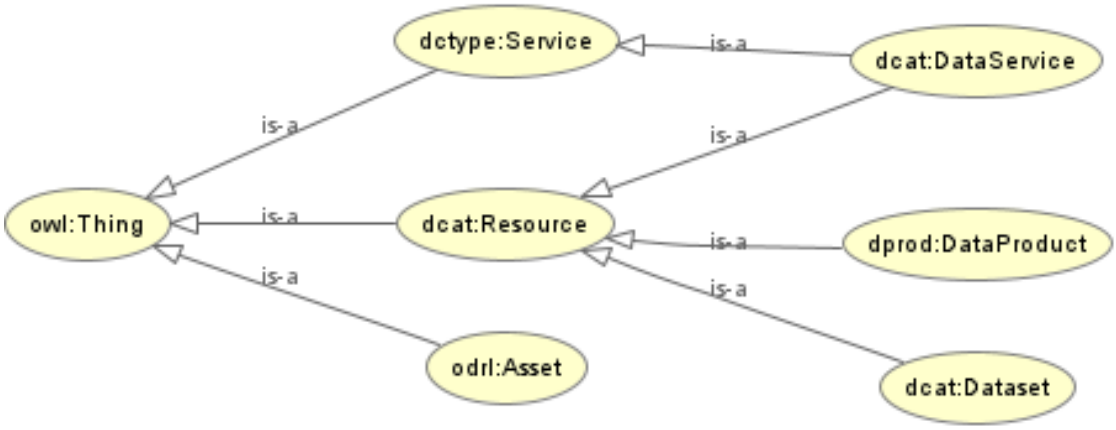

Figure 5.4: Class hierarchy of Data Product Module through the OWLViz plugin in Protégé; prefixes denote the reference ontology

is captured through dprod:DataProduct, with its interfaces represented as Data Services and its payload as a Dataset. Two specializations adapt these standards to the local data mesh domain: the input and output ports are modeled as dcat:DataService (linked via dprod:inputPort/dprod:outputPort), and the exposed dataset is connected through dprod:outputDataset.

Afterward, the data product module was expanded and simplified. The lineage and protocol simplifications are described in the Ontology Modifications and Extensions section (Section 5.2) and are applied here without repetition.

- Data lineage is simplified to reduce graph traversal complexity. Instead of the standard dprod lineage paths, dependencies are mapped directly between interface entities. Specifically, an input port (dcat:DataService) is linked directly to an upstream output port (dcat:DataService) using the Dublin Core property dct:source.
- Because standard vocabularies lack a formal protocol class, communication protocols are simplified as literal strings using the dprod:protocol data property applied to the dcat:DataService.

- **Example Scenario**
  To support downstream commission calculations (i.e., dprod:purpose), the "Sales Employee Performance" data product (i.e., dprod:DataProduct), situated within the "Sales" business domain (i.e., dprod:domain), aggregates order margins (i.e., dcat:Dataset) and exposes this dataset exclusively through a PostgreSQL output port (i.e., dcat:DataService accessed via dprod:outputPort). This specific output port acts as the technical interface (defined by dprod:protocol and dcat:endpointURL) and the downstream BI Dashboarding services (i.e., a downstream dcat:DataService representing an input port) can connect to this data product (i.e., dct:source).

- **Diagram**

- **Building Block**
  https://github.com/TeunHoven/data-contract-ontology/blob/main/datacontract_product.ttl

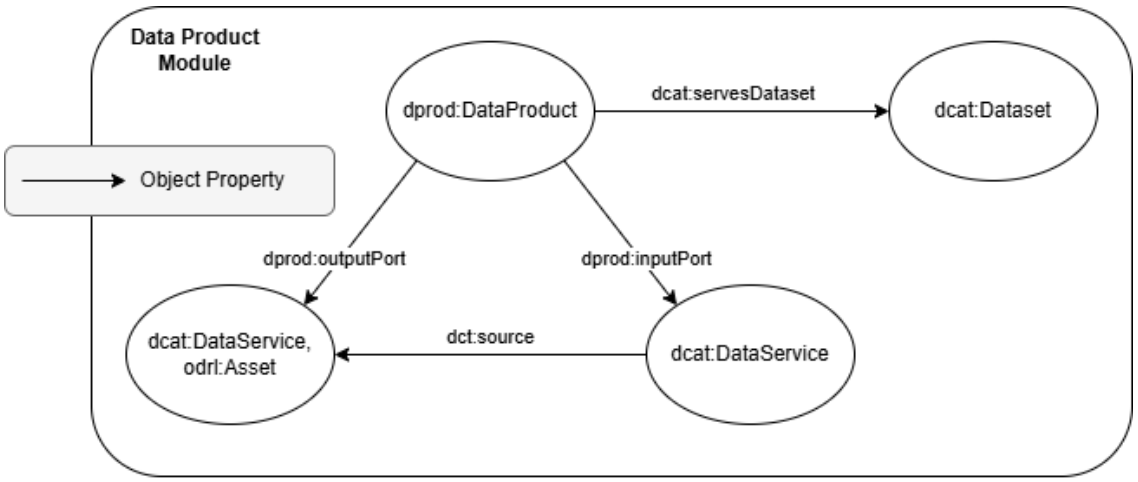

Figure 5.5: Data Contract Ontology data product module; prefixes denote the reference ontology; prefixes denote the reference ontology

5.2.5.2 Ownership Module

- **Intent**
  To model the organizational structure of the enterprise and track the temporal ownership of the data product.
- **Competency Questions**
  CQ6, CQ7, CQ8, CQ9
- **Elements**
  The classes and properties of this module, together with their hierarchy, are depicted in Figure 5.6.

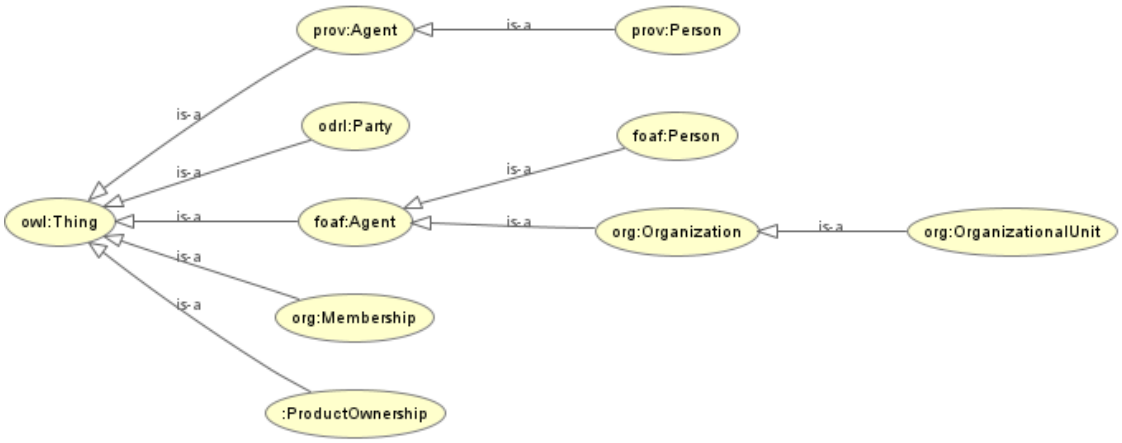

Figure 5.6: Class hierarchy of Ownership Module through the OWLViz plugin in Protégé; prefixes denote the reference ontology

- **Reengineered From**
  The core of the ownership module is derived from the W3C Provenance Ontology (PROV-O) and the W3C Organization Ontology (`org`). We mapped these standards to the data mesh operating model:

  - Individual stakeholders and employees are `prov:Person` and `foaf:Person`, representing agents inside the organization. For the policies, `odrl:Party` is also used to describe a person as this is the assigner of the data contract (see Section 5.2.5.3).
  - Teams within an organization are modeled as `org:OrganizationalUnit`. A team could also be the assigner of a data contract; therefore, this entity needs `odrl:Party`.
  - The relationship between a person and a team is defined by the class `org:Membership`. We can add additional information such as the role, the start and end dates, and a description of the membership.

To provide lineage of data product ownership, we extended the standard ontology with the class `ProductOwnership`. This class connects the data product (through `dataProduct`) and the owner (through `dprod:dataProductOwner`). To support a history of product ownership, the `prov:startedAtTime` and `prov:endedAtTime` properties are applied directly to the `ProductOwnership` class, ensuring a lineage of ownership.

- **Example Scenario**
  Within an enterprise, John Doe (i.e., `prov:Person`) is employed as a data product owner role (i.e., `org:role`) and is part (i.e., `org:Membership`) of the "HR Data Analytics Team" (i.e., `org:OrganizationalUnit`). As data product owner of a data product (i.e., `dcont:ProductOwnership`) that links Alex to the Sales Employee Performance data product (i.e., `dprod:DataProduct`). This ownership has started on January 15, 2026 (i.e., `prov:startedAtTime`), establishing a temporal provenance record that tracks who is responsible for the data product.

- **Diagram**

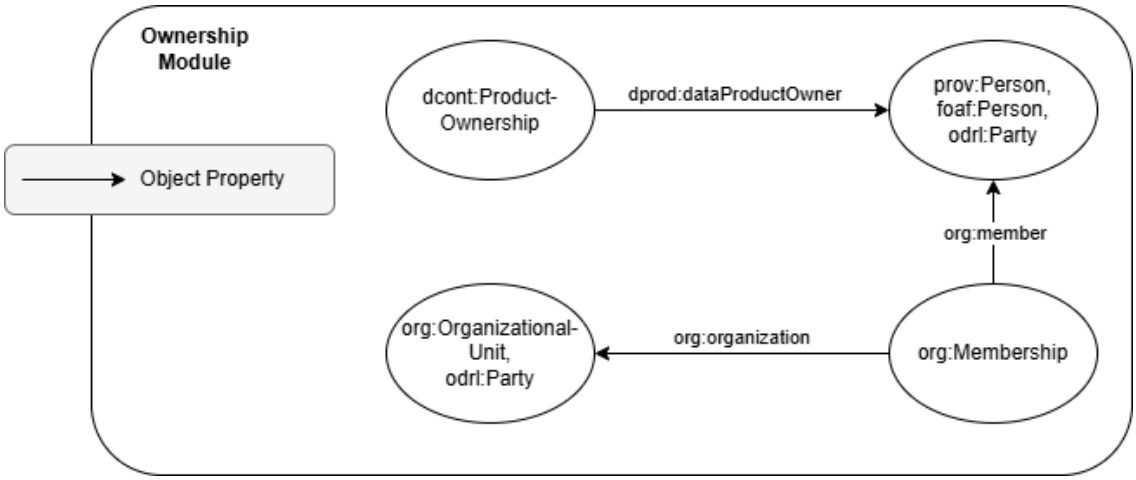

Figure 5.7: Data Contract Ontology ownership module; prefixes denote the reference ontology

- **Building Block**
  `https://github.com/TeunHoven/data-contract-ontology/blob/main/datacontract_ownership.ttl`

5.2.5.3 Data Contract Module

- **Intent**
  Representing the data contract agreements, such as SLAs, permitted usage contexts, and governance constraints that apply to a data product's output port.
- **Competency Questions**
  CQ18, CQ19, CQ20, CQ21, CQ22, CQ23, CQ24, CQ25, CQ26, CQ27, CQ28, CQ29
- **Elements**
  The classes and properties of this module, together with their hierarchy, are depicted in Figure 5.8.

Figure 5.8: Class hierarchy of Data Contract Module through the OWLViz plugin in Protégé; prefixes denote the reference ontology

- **Reengineered From**
  The core of the data contract module is derived heavily from the W3C Open Digital Rights

(ODRL). ODRL provides a model for defining policies, rules (permissions, prohibitions, and duties), and constraints. To adapt the ODRL ontology to the data mesh specification, we performed the following adaptations:

  - We created a specialized `odrl:Policy` class for the data contracts: `DataContractPolicy`.
  - Since the new policy class is specifically for an output port, we restrict the target (`odrl:target`) to an output entity with classes `dcat:DataService` and `odrl:Asset`.

To accommodate more specific permissions required by modern data practices, we extended the ontology with usage terms. The terms are of class `odrl:Action` and include `ensure` for SLA guarantees and `write`, `distribute`, `use`, `use_masked`, `read`, and `read_masked` for privacy-preserving data consumption.

- **Example Scenario**
  The HR Data Analytics Team (i.e., `odrl:Party` acting as the `odrl:assigner`) enforces a data contract (i.e., `DataContractPolicy`) on the Postgres output port (i.e., the `odrl:target`). This policy contains a permission rule (i.e., `odrl:Permission`) that allows automated transformation tools to write data (i.e., `write`), and grants read access (i.e., `read`) to the Sales Leadership role (i.e., `odrl:assignee`). Simultaneously, the policy includes a prohibition rule (i.e., `odrl:Prohibition`) explicitly preventing external distribution (i.e., `distribute`), and imposes a binding duty (i.e., `odrl:Duty`) ensuring (i.e., `ensure`) 95% endpoint availability via a measurable performance constraint (i.e., `odrl:Constraint`).

- **Diagram**

- **Building Block**
  `https://github.com/TeunHoven/data-contract-ontology/blob/main/datacontract_contract.ttl`

5.2.5.4 Dataset Schema Module

- **Intent**
  The dataset schema module represents the underlying dataset and establishes the data quality rules and guarantees.

- **Competency Questions**
  CQ10, CQ14, CQ15, CQ16, CQ17

- **Elements**
  The classes and properties of this module, together with their hierarchy, are depicted in Figure 5.10.

- **Reengineered From**
  This module is constructed by combining the W3C CSV on the Web (CSVW) ontology, the W3C Data Quality Vocabulary (DQV), and the Data Privacy Vocabulary (DPV). The dataset schema topology is defined as follows:

  - The `csvw:Schema` class is utilized to represent the tabular structure of a dataset, with data fields defined as `csvw:Column`.
  - Database constraints (e.g., nullability, data types, and primary keys) are captured natively using CSVW's built-in datatype properties, specifically `csvw:required`, `csvw:datatype`, and `csvw:primaryKey`.

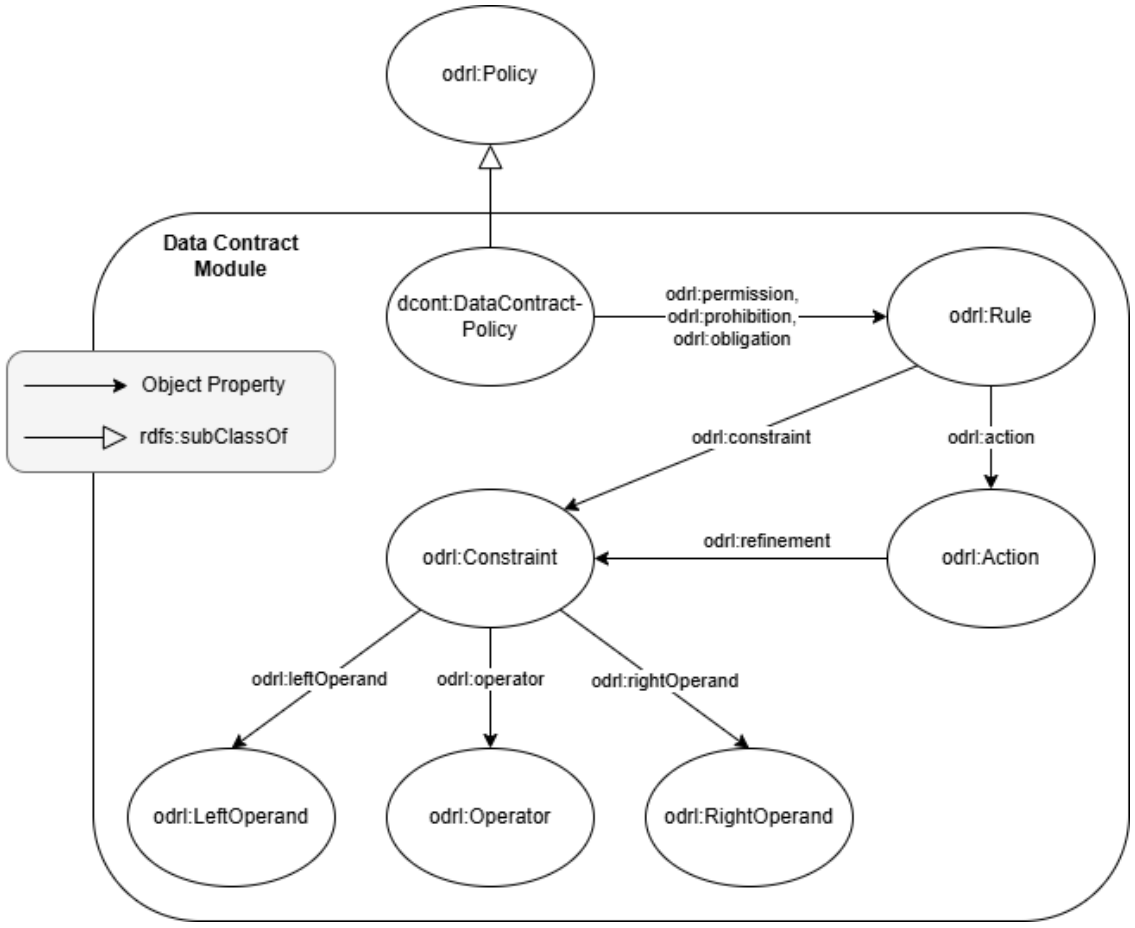

Figure 5.9: Data Contract Ontology data contract module; prefixes denote the reference ontology; prefixes denote the reference ontology

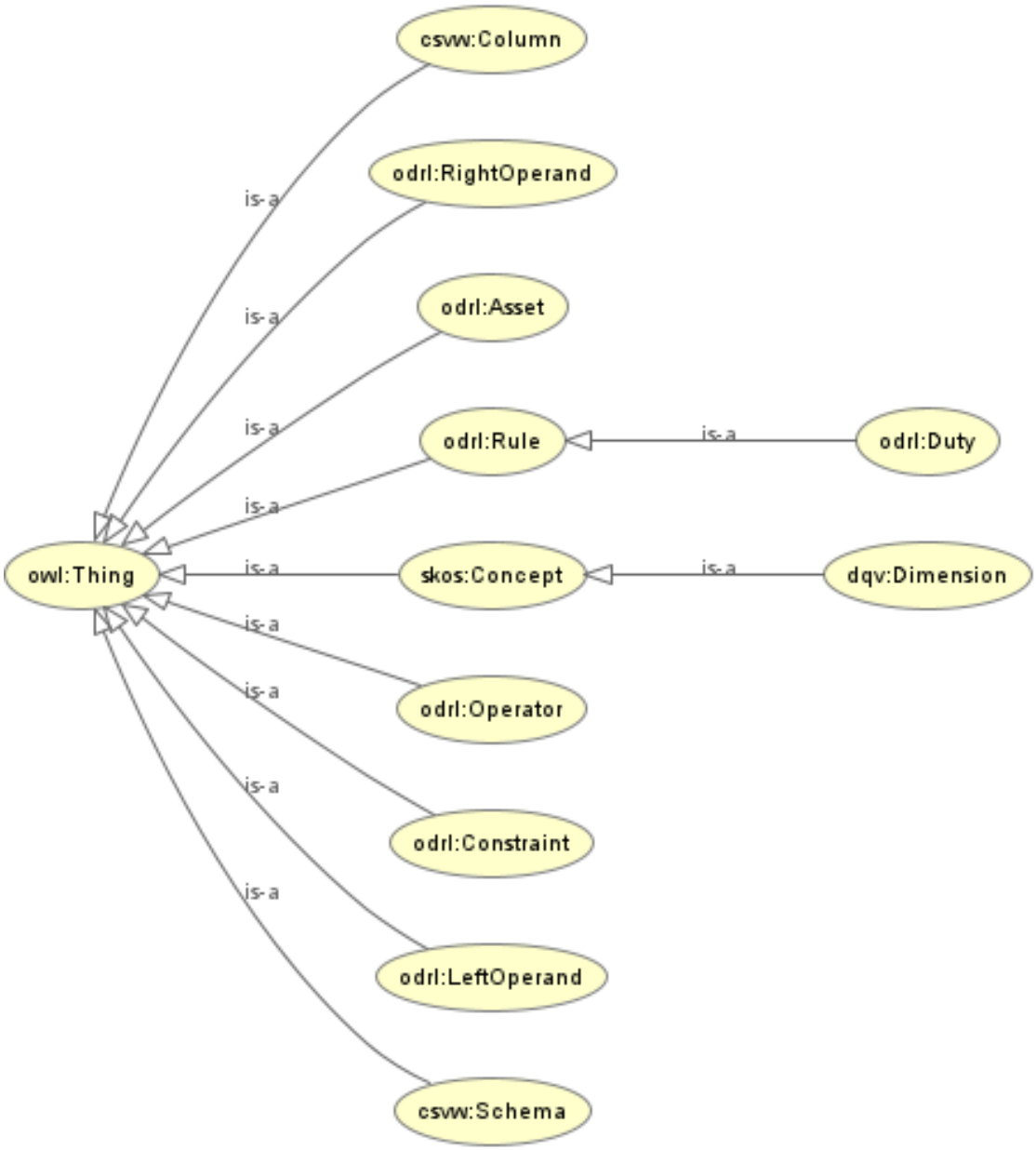

Figure 5.10: Class hierarchy of Dataset Schema Module through the OWLViz plugin in Protégé; prefixes denote the reference ontology

To allow governance information in the dataset schema module, the ontology was expanded with quality and privacy axioms:

- To prepare the schema for formal SLAs, the DQV is integrated. Standardized ISO 25012 data quality characteristics[11] (e.g., Accuracy, Completeness, Consistency) are modeled as `dqv:Dimension` instances under the custom `iso:` namespace. The entities in this namespace are classified as both `dqv:Dimension` and `odrl:LeftOperand` such that it is easier to create constraints on the columns.
- To support automated privacy governance (e.g., GDPR compliance), the DPV is incorporated. This allows specific `csvw:Column` instances to be annotated with privacy-related classes, such as `dpv:PersonalData` via `dcat:theme`, thereby triggering downstream access prohibitions.

- **Example Scenario**
  The underlying dataset schema (i.e., `csvw:Schema` connected via `dct:conformsTo`) strictly defines the "EmployeeName" column (i.e., `csvw:Column`) as required (i.e., enforced via the `csvw:required` and `csvw:datatype` datatype properties). Because this column contains sensitive Personally Identifiable Information (PII), it is annotated as personal data (i.e., assigned the class `dpv:PersonalData`). Consequently, its quality is strictly regulated: the contract guarantees that the Completeness dimension (i.e., `iso:Completeness` acting as an `odrl:LeftOperand`) applied to this specific target permits a maximum of only 0.1% missing values to account for system delays.

- **Diagram**

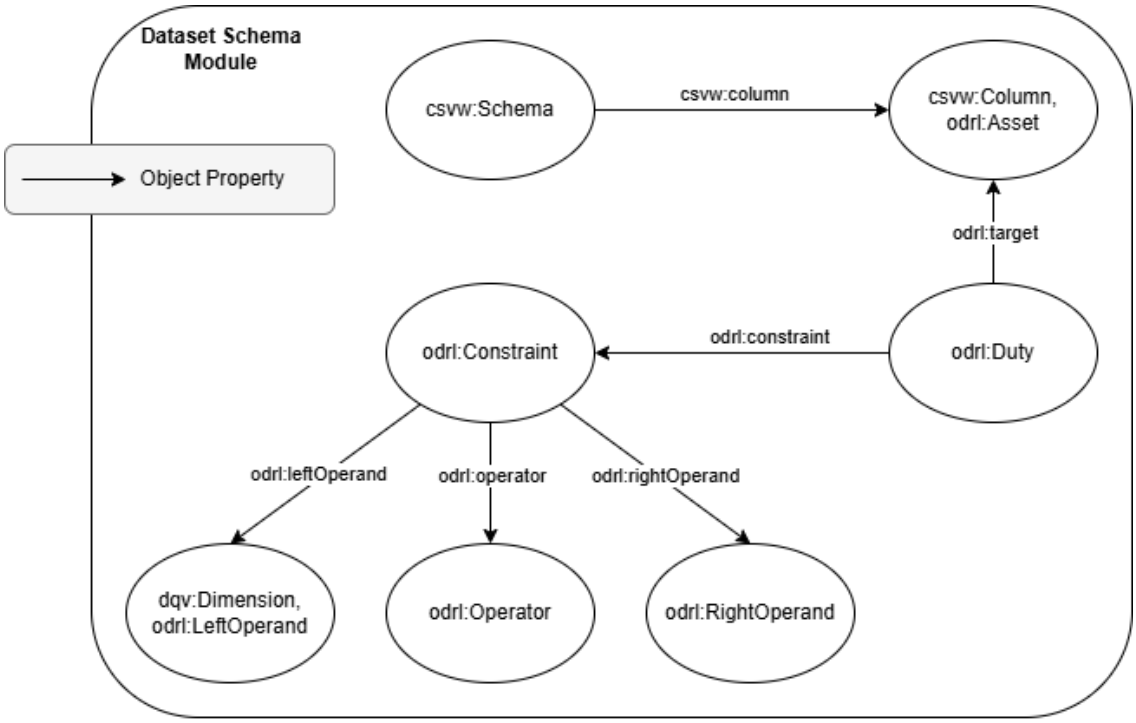

Figure 5.11: Data Contract Ontology dataset schema module; prefixes denote the reference ontology

[11] https://iso25000.com/index.php/en/iso-25000-standards/iso-25012

- **Building Block**
  https://github.com/TeunHoven/data-contract-ontology/blob/main/datacontract_schema.ttl

5.2.5.5 Provenance Module

- **Intent**
  To track the generation history and source of the data contracts parsed into the KG by the LLM, ensuring full system auditability.

- **Competency Questions**
  CQ21

- **Elements**
  The classes and properties of this module, together with their hierarchy, are depicted in Figure 5.12.

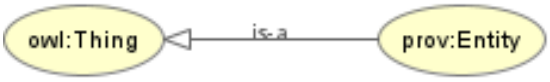

Figure 5.12: Class hierarchy of Provenance Module through the OWLViz plugin in Protégé; prefixes denote the reference ontology

- **Reengineered From**
  The provenance module is implemented directly by the W3C Provenance Ontology (PROV-O). The Data Contract Ontology utilizes a simplified, entity-centric provenance pattern to improve the auditability of the KG:

  - Every parsed data contract instantiated by the LLM is classified as a generic `prov:Entity`.
  - To bridge the gap between the source and the KG, the `prov:hadPrimarySource` object property is utilized. This links the generated RDF entities back to the data contract that the LLM ingested.
  - The timestamp of the LLM inference is captured using the `prov:generatedAtTime` datatype property, and this allows for time-based versioning.

- **Example Scenario**
  During the ingestion phase, the generated RDF entities (i.e., `prov:Entity`) defining the Sales Employee Performance contract were generated by the framework on May 18, 2026 (i.e., `prov:generatedAtTime`). To ensure auditability, the entities are linked to their originating artifact, the ODCS-based semi-structured `sales_contract.yaml` file (i.e., linked via `prov:hadPrimarySource`), establishing a lineage from human input to machine-executable logic.

- **Diagram**

- **Building Block**
  https://github.com/TeunHoven/data-contract-ontology/blob/main/datacontract_provenance.ttl

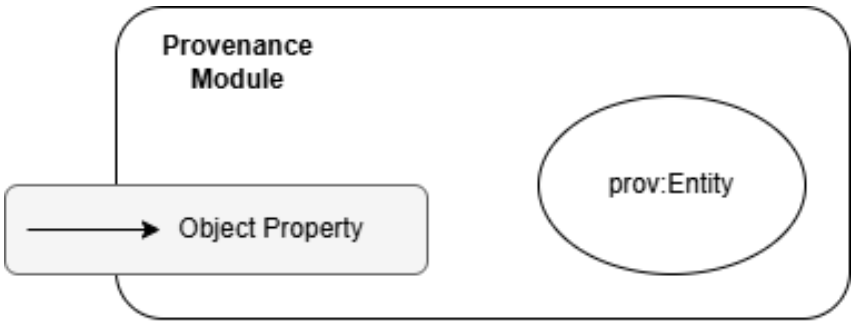

Figure 5.13: Data Contract Ontology provenance module; prefixes denote the reference ontology

5.2.5.6 Module Integration: Data Contract Ontology

- **Intent**
  By integrating the modules described in Section 5.2.5.1 through Section 5.2.5.5, we operationalize data mesh governance by linking the architectural topology of data products with their organizational ownership, dataset schema, and usage & quality rules.

- **Competency Questions**
  CQ22, CQ23, CQ24

- **Elements**
  The classes and properties of all modules, together with their hierarchy, are depicted in Figure 5.14.

- **Reengineered From**
  The Data Contract Ontology is established by integrating the four primary modules (data product, ownership, data contract, dataset schema) and the provenance submodule. The integration between the modules is achieved through object property relationships that connect the central classes of each module. These relationships are the topological glue:

  - **Data Contract to Data Product:** The data contract module acts as the governance module. The `odrl:target` object property connects the usage policy (`DataContractPolicy`) directly to the output port interface of the data product module (`dcat:DataService` and `odrl:Asset`).
  - **Ownership to Data Product:** Data product ownership is important as it shows the lineage of ownership of the data products through the extended `dataProduct` relation, linking the `ProductOwnership` entity to its `dprod:DataProduct`. Furthermore, the `dprod:dataProductOwner` links both the data product and the ownership back to the person responsible (`prov:Person`)
  - **Data Product to Dataset Schema:** The dataset exposed by the data product (`dcat:Dataset`) is bound to its schema definition (`csvw:Schema`) using Dublin Core `dct:conformsTo` property.

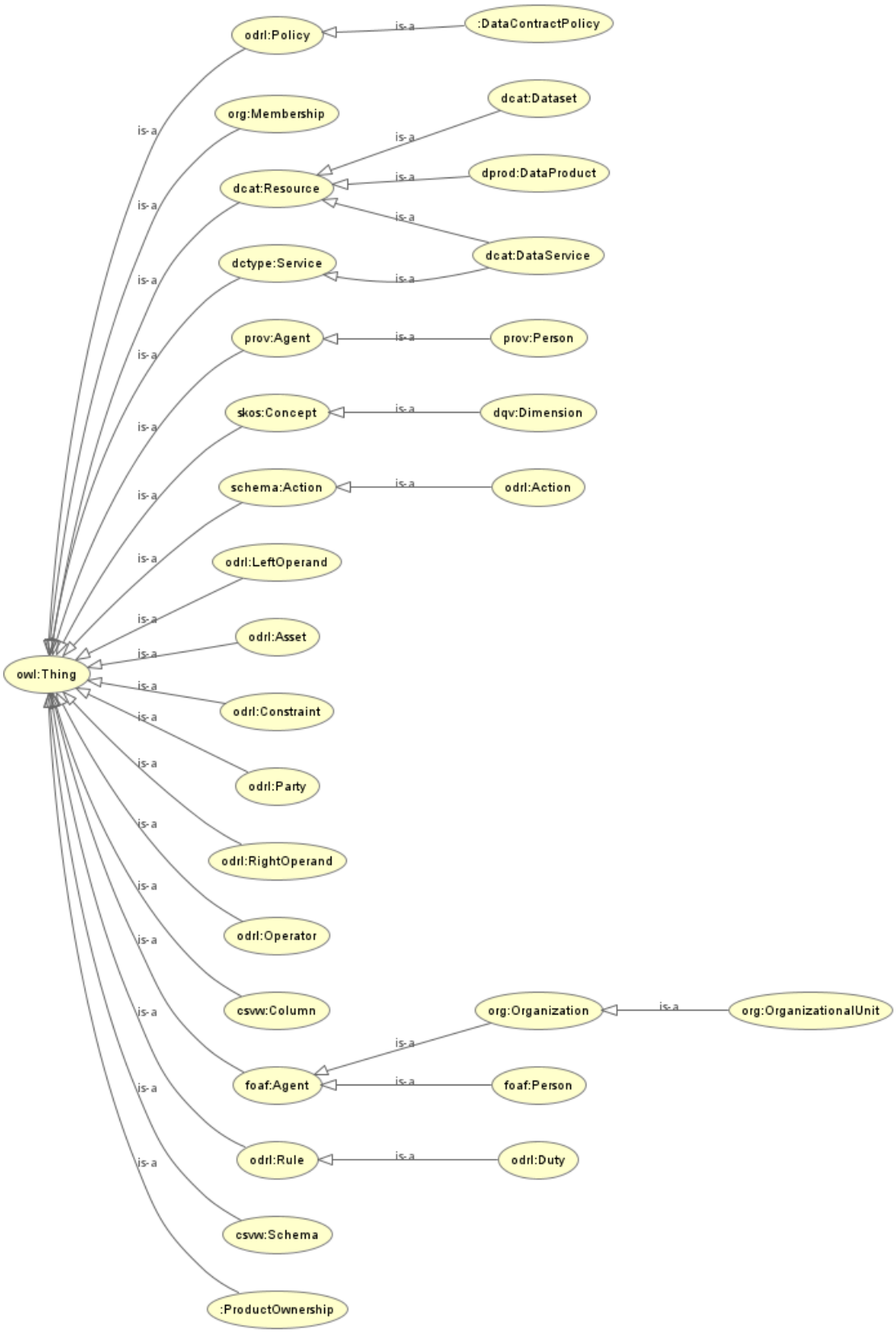

Figure 5.14: Class hierarchy of the Data Contract Ontology through the OWLViz plugin in Protégé; prefixes denote the reference ontology

  - **Provenance to All**: The provenance sub-module encapsulates the generated graph by classifying the RDF entities as a `prov:Entity`, utilizing `prov:hadPrimarySource` to link the extracted entities back to the original data contract source.

- **Example Scenario**
  The "Sales Employee Performance" data product (i.e., `dprod:DataProduct` captured in the data product module) is governed by John Doe (i.e., `prov:Person` connected via `dcont:ProductOwnership` in the ownership module). The output port of this product exposes a dataset that adheres to a tabular schema (i.e., `csvw:Schema` linked via `dct:conformsTo` in the dataset schema module). The schema has certain data quality and privacy constraints that target the sensitive "EmployeeName" column (i.e., `csvw:Column`) to enforce a strict missing-value threshold (i.e., `iso:Completeness` acting as an `odrl:LeftOperand`). To ensure compliant data consumption, the HR Data Analytics Team assigned a data contract (i.e., `DataContractPolicy` in the data contract module). This policy targets the PostgreSQL output port (i.e., `dcat:DataService` linked via `odrl:target` in the data product module) to manage access and service-level guarantees. Within this contract, specific duties (i.e., `odrl:Duty`) are defined that target the output port to enforce SLAs (i.e., `odrl:Constraints` utilizing `ensure`), while permissions (i.e., `odrl:Permission`) and prohibitions (i.e., `odrl:Prohibition`) dictate the authorized usage and distribution limits (i.e., `odrl:Action` such as `read` or `distribute`) of the served data. The entire interconnected graph representing these rules was parsed by the LLM from `sales_contract.yaml` (i.e., `prov:hadPrimarySource` in the provenance module).

- **Diagram**

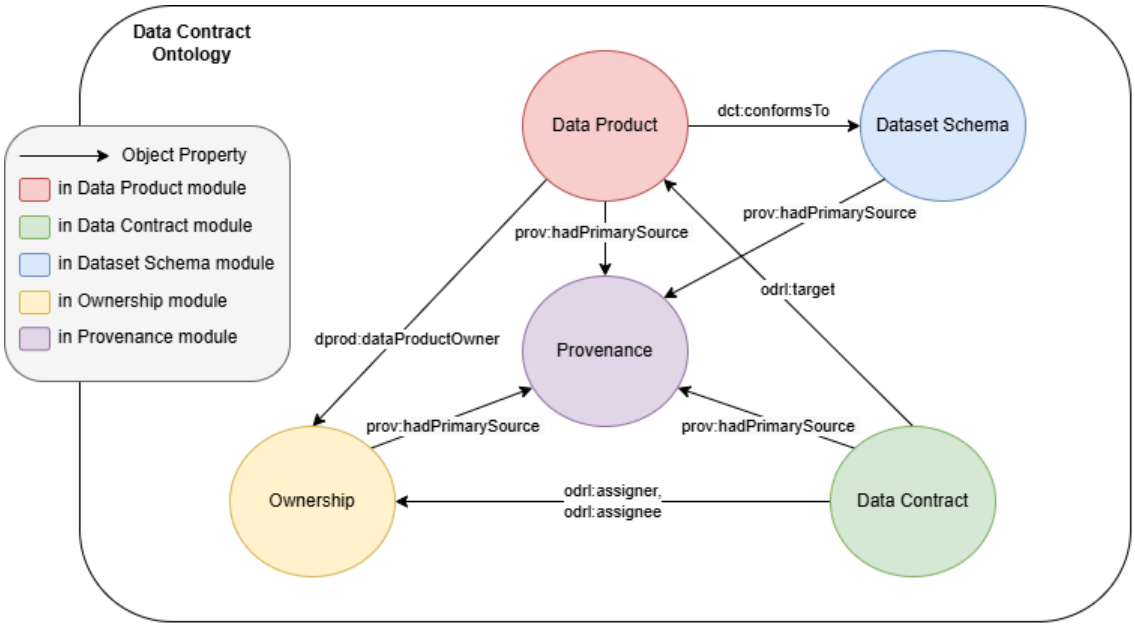

Figure 5.15: Data Contract Ontology integration of modules

- **Building Block**
  `https://github.com/TeunHoven/data-contract-ontology/blob/main/datacontract_ontology.ttl`

5.3 Knowledge Graph Generation

To facilitate the generation of the KG, the *Mistral Large 3*[12] model from Mistral was selected. Because the exploratory interviews were conducted with predominantly European enterprise participants, and this research is itself situated in a European context, data sovereignty emerged as a recurring priority. The Mistral model aligns with this setting as a European provider whose licensing supports regional data-governance requirements. The decisive factor, however, is the model's open-source nature under an Apache 2.0 license, which makes the framework reproducible and self-hostable. The absolute predictive performance of the language model is not the primary focus of this research: the framework is model-agnostic, and Mistral Large 3 serves as a representative, openly available instantiation rather than a performance-optimal choice. Its extraction performance nonetheless matters and is evaluated in Chapter 6.

All extractions were run with a low-variance configuration to align with the determinism requirement of the framework. The temperature was set to 0.1, following the Mistral's recommendation to keep the temperature low (below 0.1) for production and daily-driver use[13]. Setting the temperature to a low value makes the output more conservative and repeatable as it increases the distribution towards the most probable tokens. Top-p (nucleus) sampling instead restricts the choice at each step to the smallest set of tokens whose cumulative probability reaches p, discarding the tail of unlikely tokens [74]. Because the framework requires a reproducible foundation rather than diverse output, the top-p variable was set to 0.5. Both values were held this low during all runs, but this does not eliminate run-to-run variance [74].

The complete sequential workflow for generating the RDF triples from the source data contracts is visualized in Figure 5.16.

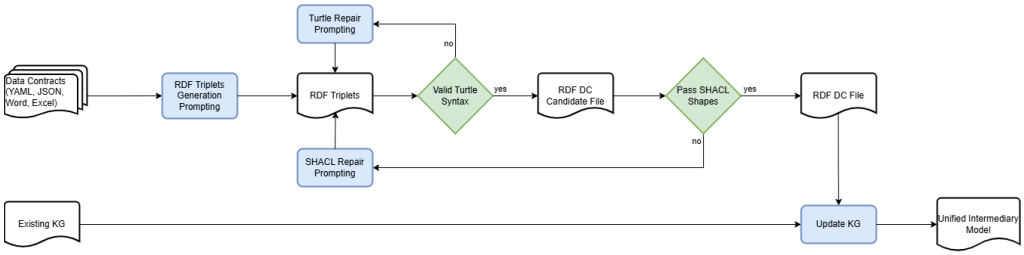

Figure 5.16: RDF Knowledge Graph Generation Workflow

To ensure the output is complete and well formed, syntactic errors (e.g., malformed Turtle) are routed to a Turtle repair-prompting loop before SHACL validation starts, and structural violations (e.g., SHACL constraint errors) are routed to a corresponding repair loop, so that the final result matches the target syntax and structure. This repair stage applies to both extraction strategies.

5.3.1 RDF Triple Extraction

Two extraction strategies for generating the RDF triples were designed, implemented, and evaluated within a single design iteration. The first strategy establishes a baseline of the LLM's extraction capabilities by prompting to output all RDF triples. The second separates the task into two sequential phases, entity extraction and relationship extraction. Both strategies are evaluated against the same data contracts, allowing the two strategies to be equally assessed.

[12] https://huggingface.co/mistralai/Mistral-Large-3-675B-Instruct-2512
[13] https://huggingface.co/mistralai/Mistral-Large-3-675B-Instruct-2512

5.3.1.1 Direct RDF Triple Generation

In the first strategy, the framework relies on the LLM to directly execute entity and relationship mapping in a single pass. As illustrated in Figure 5.17, the LLM is instructed to parse the data contract, identify critical entities (e.g., Data Products, Stakeholders, SLAs), and format them directly into RDF triples. While this direct generation approach establishes a functional proof of concept and demonstrates the model's baseline semantic understanding, unconstrained LLM outputs remain susceptible to minor structural hallucinations and syntactic deviations from the target ontology.

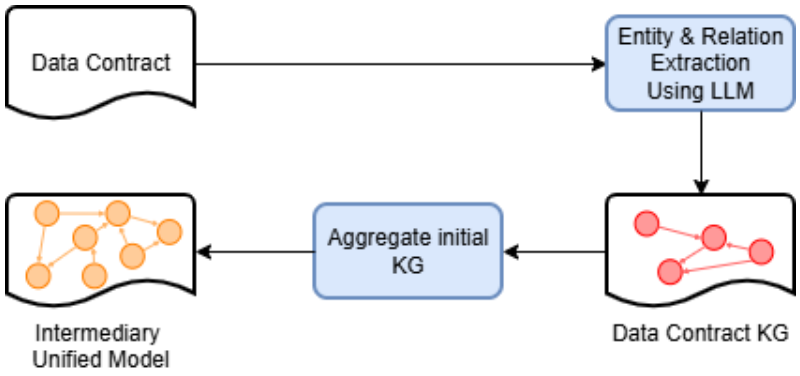

Figure 5.17: Direct translation of data contracts into RDF triples

5.3.1.2 Split Entity-Relationship Extraction

In the second strategy, as visualized in Figure 5.18, the process is divided into two sequential stages. Decomposing the task is expected to improve correctness compared to direct generation by reducing the search space. By first resolving which entities exist, the relation extraction stage operates over a closed set of known subjects and objects rather than generating both simultaneously. This effectively reduces the search space in the relation extraction step, which may affect the performance of the overall extraction by the LLM.

Crucially, both the entity and relationship extraction pipelines employ strict constraint prompting to ensure interoperability. By constraining the model with a formal JSON schema (provided in Appendix C), the language model is deterministically constrained to output the knowledge as a validated JSON object, which is then deterministically serialized into RDF. This bridges probabilistic text generation and the requirements of downstream KG synthesis: malformed output is caught at schema validation rather than inserted into the resulting graph.

5.3.2 Prompt Techniques

The application of LLMs to KG synthesis is a highly promising approach that requires careful evaluation of model-specific prompting strategies. Recent research by Martin et al. compared various prompt engineering techniques across different models [70]. Their findings demonstrate that for Mistral-based architectures (specifically the Mistral 7B model), One-Shot Prompting combined with Chain-of-Thought reasoning yields optimal extraction performance [70]. This contrasts with the behavior observed in other models (DeepSeek-R1 8B, Gemma 3 4B, Llama 3.1 8B, NuExtract 2.0 8B, and Qwen 2.5 7B), where Few-Shot Prompting proved to be the most effective strategy.

To systematically extract the necessary entities and relations for the KG synthesis, this framework evaluates the following prompt engineering strategies:

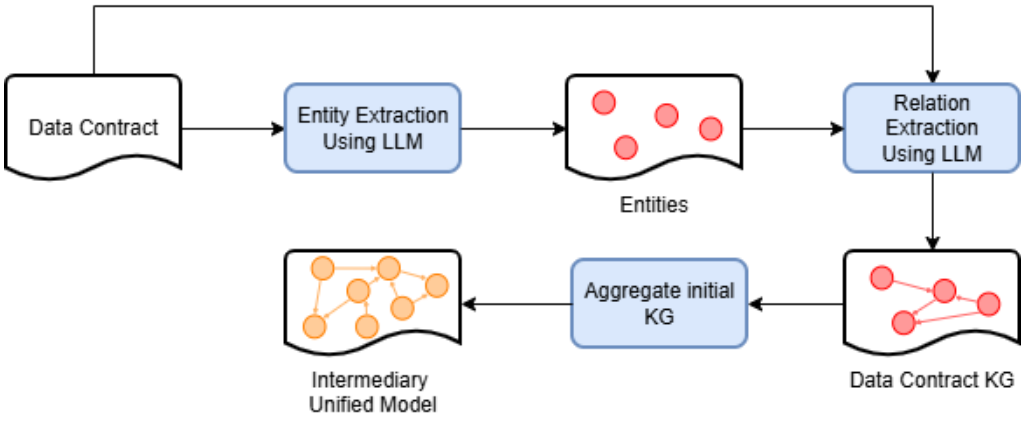

Figure 5.18: Process flow for the split entity-relationship extraction pipeline

- **Zero-Shot Prompting (ZSP)**: In this approach, the model is provided with a natural language description of the task, without any labeled data or specific input-output examples at inference time [17, 86]. The model must rely entirely on its pre-existing knowledge gained during training to generate the correct entities and relations. The final zero-shot prompt can be seen in Appendix D.2.
- **One-Shot Prompting (OSP)**: This technique extends zero-shot prompting by providing the model with exactly one demonstration of the task (a single input-output pair) alongside the task description [17]. The example (see Appendix D.3) helps the LLM with understanding the task, its context, and the output format. While an example improves model performance on complex tasks, it also requires more tokens, which may be costly and can introduce a bias toward specific words or their behavior [86].
- **Chain-of-Thought Prompting (CoT)**: COT prompting improves the model's ability to perform complex tasks by instructing it to generate a series of intermediate steps before outputting a final answer [104]. In our prompt (see Appendix D.4), we instructed the LLM to conduct its reasoning internally via a hidden chain of thought, ensuring that its final output consists exclusively of RDF [70].
- **One-Shot with Chain-of-Thought Prompting (OSP & CoT)**: This hybrid strategy combines the learning benefits from the OSP with the reasoning benefits from the CoT. The model is provided with a single example, just like the OSP, but also with instructions to perform intermediate reasoning, similar to CoT, as shown in Appendix D.5. This combination reduces the number of errors and improves the OSP technique [86].

Regardless of the prompt technique, the base input remains consistent across all evaluations. We start by retrieving the ontology defined in Section 5.2 from the graph database, such that additions are dynamically extracted instead of hard-coded. Then we parse the data product ID from the data contract and match it to its corresponding entity in the KG. This allows us to extract all currently available entities and relations connected to the data product. These graph extractions are then added to the prompt, along with the raw data contract specifications.

Depending on the specific pipeline strategy, this input is processed further. In the entity and relation extraction pipeline, the process runs sequentially: the prompt used for relation extraction is extended to include the entities identified during the first extraction step.

5.4 Architecture Integration

The transition of the LACE Framework from an isolated extraction pipeline to a component of a decentralized ecosystem requires defining its integration within the enterprise architecture. This section outlines how the LACE Framework is positioned in a data mesh environment.

5.4.1 Semantic Module

At the core of the proposed integration of the LACE Framework is an ontology layer powered by a semantic KG called the Semantic Module. This module serves as the central repository for metadata, domain context, and governance rules. The LACE Framework helps populate this module by ingesting the data contracts written by data product developers and extracting the necessary information to be stored in the KG. Figure 5.19 illustrates how the Semantic Module interacts with the rest of the data mesh.

Many data products can exist in a distributed environment, and each lives in a data product container with the data product, metadata, and its managing agents and services [42]. We separate the three types of communication into input, output, and management interfaces. The first interfaces are for data exchange: the input interface receives data from source systems or other data products, while the output interface provides data to consumers or other data products. Finally, the management interface enables monitoring and control of the behavior of data products (e.g., monitoring performance, routing data, and enforcing policies and data contracts) [42]. The management interface directly communicates with the management and governance plane to apply new configurations and policy updates to the data products' containers. This plane initially consists of the data mesh dashboard, policy authoring service, and policy repository, which interact with the Semantic Module of our solution.

With the Semantic Module as a core component, the architecture is one step closer to being an automated data mesh. By operationalizing this intelligence, a knowledge catalog is created to generate actionable insights from regulations, business glossaries, and active metadata. These insights can be used to adjust configurations or to notify data product owners of misalignments between governance rules or regulations and data products [48].

Another advantage of integrating the Semantic Module into the data mesh environment is that it improves any existing data catalog. By using open initiatives, such as ontologies, this approach avoids vendor lock-in and enhances interoperability among data management tools. Additionally, adopting an ontology as a metadata layer is highly recommended as it is the most expressive data model and provides built-in capabilities for consistency checking, inferencing of new classes (reasoning), and predictive analytics [29, 33].

5.4.2 LACE Framework Pipeline

To operationalize this continuous ingestion into the Semantic Module, the LACE Framework is seamlessly embedded into the data mesh's infrastructure. The LACE Framework is positioned within the self-serve data platform layer of the Data Mesh. An example architecture of a data mesh platform is illustrated in Figure 5.20. Specifically, LACE serves as an operator application that orchestrates various tools within the ecosystem and manages the lifecycle of data products [106]. It connects decentralized domains to central governance by providing APIs for both the data product developer experience plane and the mesh experience plane [106]. By centralizing the LACE Framework into the infrastructure utility plane, it abstracts the complexities of LLM orchestration away from domain developers, while allowing them to insert their data contracts into the Semantic Module [98].

The operationalization of this extraction interface is done through automated Continuous Integration and Continuous Deployment (CI/CD) workflows and DataOps practices. When a team deploys a new data contract to the platform or the version-controlled repository,

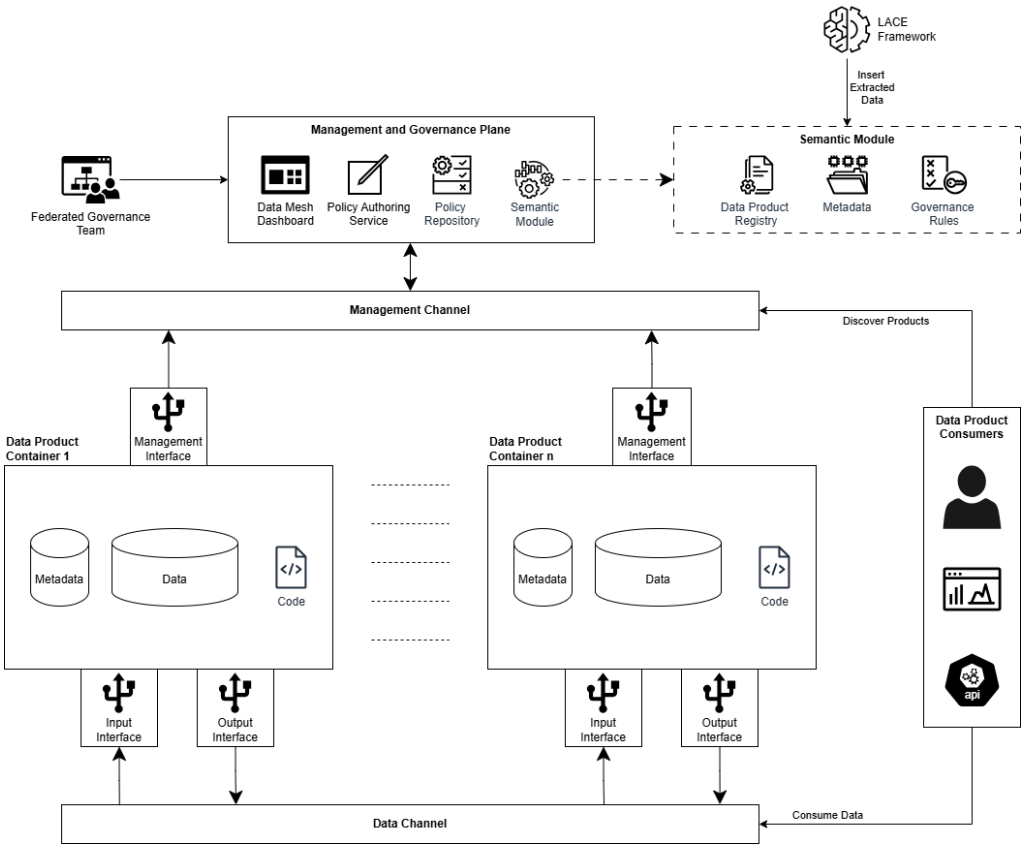

Figure 5.19: Data Mesh Runtime Reference Architecture (adapted from Goedegebuure et al. [42]). The Semantic Module sits in the management and governance plane and interacts with the mesh for management, automation, and discovery.

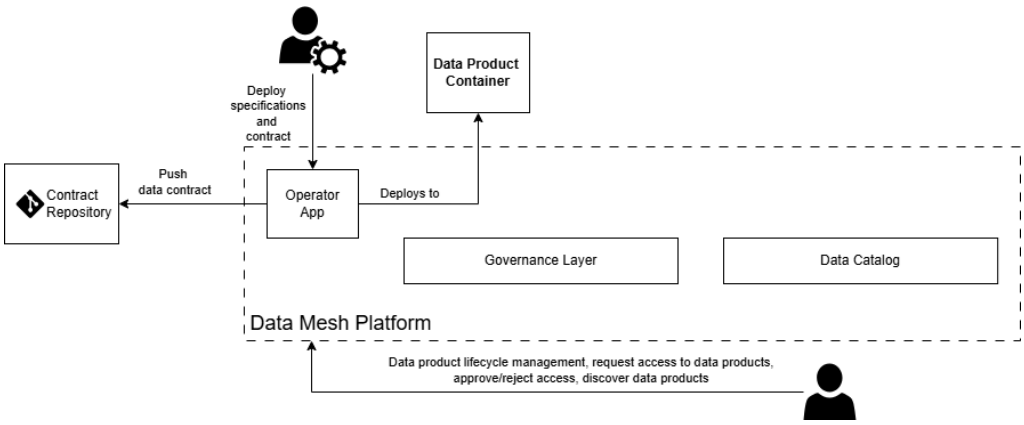

Figure 5.20: Example architecture of a data mesh platform (based on Wider et al. [106])

the CI/CD pipeline automatically triggers a series of actions [77]. This initiates the ingestion phase, where the newly updated documents are parsed within the LACE Framework and, finally, the information is inserted into the Semantic Module.

As a result, the LACE Framework populates the governance layer and the data catalog via the Semantic Module, using its generated RDF triples, as illustrated in Figure 5.21. The governance and data catalog, respectively, maintain federated business rules and facilitate the discoverability of the data products [106, 97].

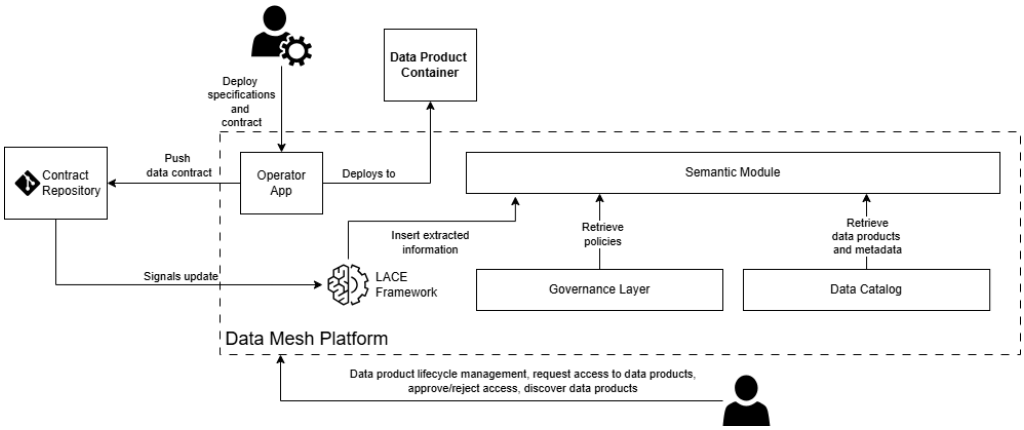

Figure 5.21: Architecture of a data mesh platform with the LACE Framework (based on Wider et al. [106])

Chapter 6

Artifact Evaluation

6.1 Evaluation Environment

To evaluate KG generation both intrinsically and extrinsically, we create an evaluation environment in which we control the data products. To make this environment as practical as possible, we will create data products where the source comes from the transactional (OLTP) database of Microsoft's Adventureworks[1]. This sample database represents a fictitious, multinational bicycle parts manufacturer and holds a hierarchy of nearly 300 employees, 500 products, 20000 customers, and 31000 sales across 68 tables in 5 schemas: *HumanResources*, *Person*, *Production*, *Purchasing*, and *Sales*. We combine the HR and Person schemas into 1 operational plane service called *HR Service*.

Because this sample database is an OLTP database, we will not use the tables directly as-is. Data products often represent analytical data; therefore, we design them accordingly [81]. To mimic a realistic data mesh architecture, we first define source-aligned data products, which extract and expose data directly from the OLTP source. The next layer consists of domain-aligned data products, which ingest data from either source-aligned or other domain-aligned products. In this layer, information is combined and transformed to generate specific business value for the domain and prepare it for downstream consumption. Finally, the consumer-aligned data products serve as the endpoints of this architecture. These products can include specialized datasets optimized for dashboards, feature stores for machine learning models, or APIs designed to serve end-user applications.

To establish a controlled environment for our evaluation, we designed a minimal mesh of data products, illustrated in Figure 6.1. This scoping limits overall complexity, enabling rigorous intrinsic and extrinsic evaluation of the Knowledge Graph's lineage-tracing capabilities without the noise of a full enterprise deployment. The implemented architecture mirrors a cross-domain ecosystem by deploying nine data products distributed across the three architectural layers: four source-aligned, three domain-aligned, and two consumer-aligned. To keep the evaluation manageable, each data product has exactly one data contract, giving nine contracts in total. This keeps the setup simple, so we can focus on how well the framework extracts a single contract rather than on how to handle several contracts for the same product.

The source-aligned layer extracts data directly from the raw operational schemas to serve as the mesh's foundational building blocks. Its four products expose, respectively, a resolved customer identity (**Customer Profiles**), a flattened order source (**Order Details**), baseline product and cost data (**Product Catalog**), and a curated employee dimension (**Employee Profiles**).

The domain-aligned layer ingests these source-aligned products to generate aggregated business value: **Customer Lifetime Value** (historical revenue and activity per customer), **Order Profitability** (revenue, cost of goods sold, and margin per order), and **Sales Employee Performance** (total margin generated per sales employee).

The consumer-aligned layer then exposes two downstream endpoints: **Customer Margin Cohorts**, supporting cohort analysis of customer profitability, and **Promotional Audience**,

[1] https://github.com/microsoft/sql-server-samples/tree/master/samples/databases/adventure-works

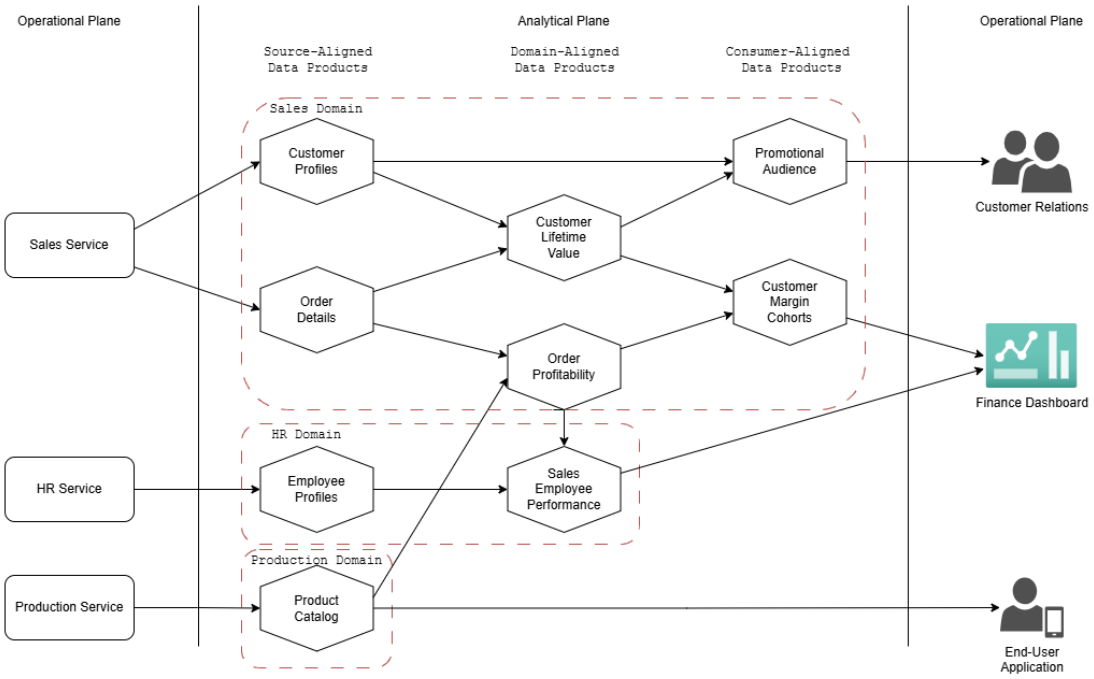

Figure 6.1: Sample Data Mesh for Evaluation

segmenting customers into engagement tiers for targeted campaigns.

By structuring the evaluation environment in this manner, we establish a chain from the OLTP AdventureWorks tables to the final analytical and operational endpoints. This showcases how cross-domain data exchange could happen in practice. Once the data contracts for these data products are parsed by the LACE Framework, this same chain is inserted as RDF triples into the Semantic Module. An initial view of the data mesh inside the Semantic Module is illustrated in Figure 6.2. The figure shows how each data product is represented as an entity with input and output ports, and how the data products connect to one another. This recreates a part of the cross-domain dependencies of Figure 6.1.

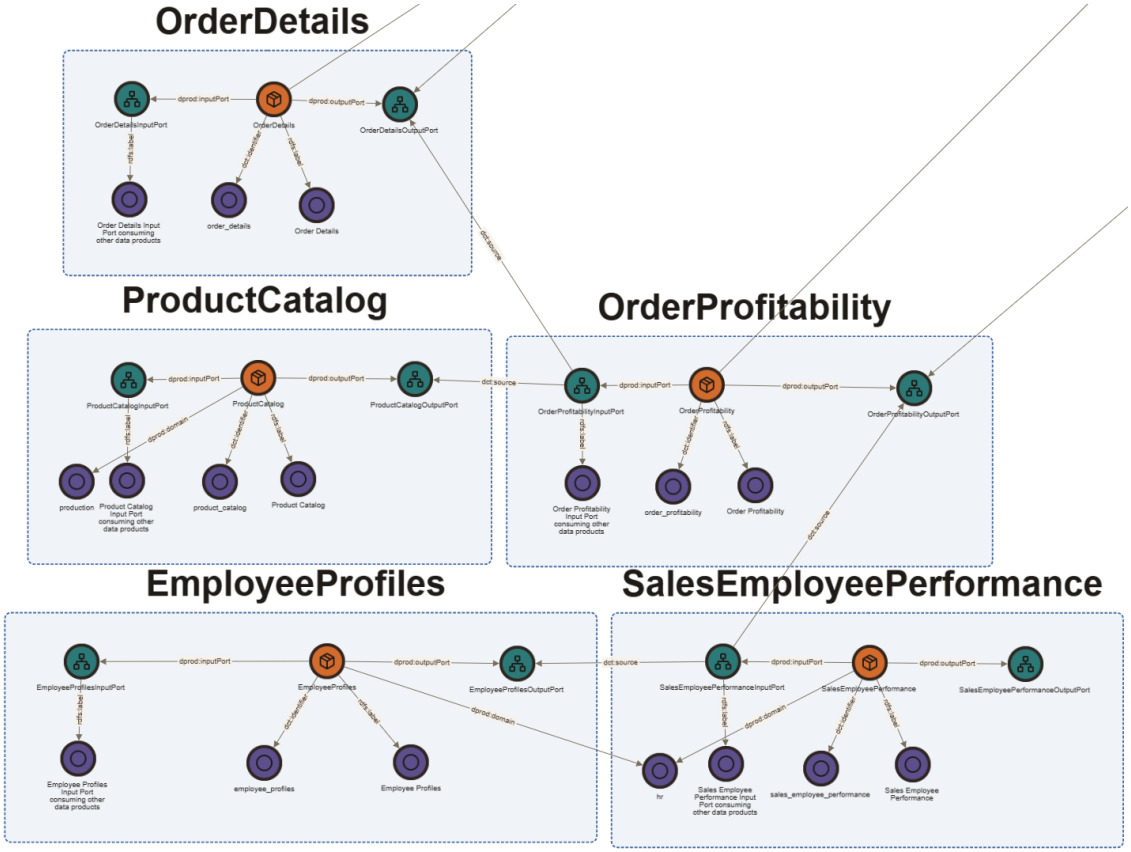

Figure 6.2: Example of data products in the Semantic Module

6.2 Ontology Evaluation

Each CQ from Section 5.2.1 was translated into a formal SPARQL query. SPARQL is the W3C standard query language for RDF and has become the de facto standard for this purpose [43]. A query specifies a graph pattern over the triples in the graph; answering it means returning every set of variable bindings for which that pattern is satisfied. As a minimal example, the query in Listing 6.1 asks for the identifier of every data product in the graph: the `WHERE` clause is a graph pattern in which `?product` and `?identifier` are variables, and the engine returns one row per binding that matches, here one identifier per data product.

```
SELECT ?identifier WHERE {
  ?product a dprod:DataProduct ;
         dct:identifier ?identifier .
}
```

Listing 6.1: A minimal SPARQL query retrieving the identifier of every data product in the graph.

A CQ is therefore considered answerable when its query returns a non-empty, correct set of bindings over the gold standard graph. The queries were used, together with a use case, to evaluate the utility and completeness of the ontology.

Because the queries are evaluated against the gold standards rather than the generated output, this evaluation is only for the ontology itself. The objective of this evaluation is to assess how well the Data Contract Ontology provides the classes, properties, and relationships required to satisfy the competency questions. A CQ that cannot be expressed, or that returns no answer, indicates a gap in the ontology rather than in the extraction framework. A CQ that cannot be expressed or returns no answer indicates a gap in the ontology rather than in the extraction framework.

Additionally, the ontology was assessed using the Ontology Pitfall Scanner (OOPS) [80] based on 40 criteria related to structure, functionality, and usability. Each part is assigned a severity level: critical, important, or minor. While it is essential to address critical pitfalls, minor criteria do not compromise the ontology's integrity or cause functional issues [80].

6.3 Intrinsic Artifact Evaluation

6.3.1 Knowledge Graph Evaluation

6.3.1.1 Gold Standard Creation

To evaluate the LACE Framework on precision, recall, and F1-score, a gold standard of RDF triples was constructed for each of the nine data contracts in the evaluation environment. Each data contract was expressed in multiple contract formats, structured YAML (ODCS), text, and Excel, to evaluate the heterogeneous-input requirement (FR1, Section 4.5.1.1). The gold standards were therefore created per format rather than per contract. The annotation began with the structured ODCS contracts, where each contract was inspected and its fields mapped to their corresponding entities in the Data Contract Ontology of Section 5.2. Because ODCS is structured, this mapping is mostly deterministic, which means the researcher's bias is low and yields the most complete set of gold standards.

The gold standards for the text and Excel contracts were then derived from these ODCS gold standards. ODCS is extensive, and a data product owner can enhance the data contract with extensive metadata (e.g., descriptions and business glossaries), whereas text and Excel data contracts do not permit such metadata. For each unstructured format, triples that the format cannot express were removed from the ground truth, yielding a gold standard that

reflects only the information extractable from the contract. Each text and Excel contract was additionally reevaluated independently after pruning, and the result was checked against the corresponding gold standard to ensure it matched.

Each extraction from the LACE Framework is evaluated against the gold standards for its own source format. This is done because extractions from an Excel-based contract should not be penalized for missing triples that exist only in the richer ODCS representation. This separation allows for evaluation based on the extraction performance of the information available in the input format.

All gold standards were produced by a single annotator, so inter-annotator agreement (e.g., Cohen's Kappa) could not be calculated. Constructing gold standards this way (human-sourced), yields high quality results when the underlying graph and ontology are not overly complex [78]. While it is costly and time-consuming, it is typically done in small-scale experiments like this one. To limit researchers' bias, the RDF triples in the ground truth were constrained to the predefined Data Contract Ontology and the ODCS format, rather than to free interpretation. Each ground truth was checked against the competency questions and validated using SHACL constraints. This limitation is further discussed in Chapter 9.

The complete gold standards are available in the project repository.

6.3.1.2 Determinism

To assess the determinism of the LACE Framework, each configuration (contract format, prompt strategy, and extraction mode), was executed 10 times under identical settings. For each configuration, we report two types of stability, introduced in Section 3.5.1.2. The quantity of the extraction is evaluated through the CV of the number of generated triples, capturing how consistently the framework produces a comparable result volume. The content of the extraction is evaluated through the Jaccard index over the repeated runs in both exact and fuzzy form.

With both these measures, we can see how stable the framework is in how much it extracts and in which triples it generates.

6.3.1.3 Knowledge Graph Main Chain

The graph-entropy ratio introduced in Section 3.5.1.3 separates the entropy of the full generated graph and that of the essential subgraph. To compute the entropy ratio of the subgraph against the full graph, the essential subgraph (*main chain*) needs to be defined. The main chain isolates the predicates that carry the ontology's governance value: the data product lineage, the attribution of ownership, and the regulatory rules. Predicates that do not contribute to this chain, such as literal labels, descriptions, timestamps, etc., are excluded. This leaves only the most important entities and relations required to answer who owns the data products, where its data originates, and under which permissions, prohibitions, and obligations it may be used.

The main chain is constructed by extracting a fixed subset of predicates from the full graph. Lineage is captured through the architectural links `dprod:inputPort`, `dprod:outputPort`, and `dct:source`, which together make the path from a data product through its interface ports to its upstream dependencies. Ownership is captured via `dprod:dataProductOwner`, which relates each data product to its responsible owner. The governance rules are captured through the ODRL predicates `odrl:permission`, `odrl:prohibition`, and `odrl:obligation`, which link a data contract policy to its rules. Figure 6.3 shows the main chain within the Data Contract Ontology by a darkened outline. The highlighted path connects the data product, ownership, and the data contract modules, while the ontology constructs with the light outline fall outside of the main chain.

By extracting these predicates and the connected subjects and objects, the resulting entropy ratio measures the proportion of the graph's complexity that only exists with the

core lineage, ownership, and governance purpose without taking into account the other descriptive, schema-level metadata that enriches the graph but can not be used for governance or automation.

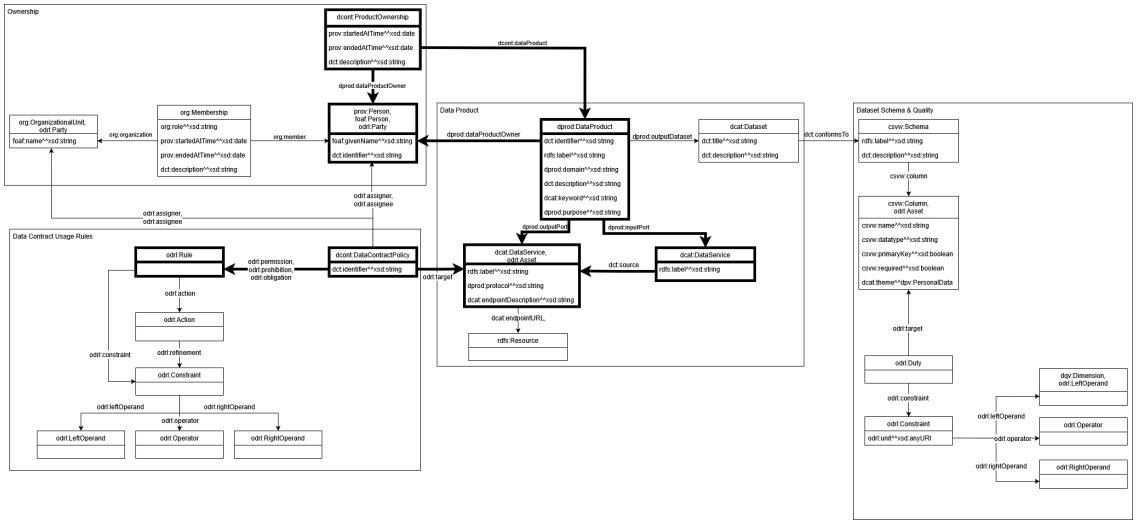

Figure 6.3: Overview of the knowledge graph main chain (dark outline)

6.4 Extrinsic Artifact Evaluation

6.4.1 Technology Acceptance Model

The extrinsic evaluation draws on the eleven industry experts who participated in the exploratory interviews, each of whom was asked during the interview about their willingness to take part in a follow-up assessment. By asking these participants, we ensured that the artifact is evaluated by practitioners with direct experience. In the end, 6 participants took part in this final evaluation round.

Each participant was presented with the controlled evaluation environment of Section 6.1, which was implemented as an interactive demo website (screenshots of this environment are provided in Appendix E). After completing a task, the participant answered the questions corresponding to the task on the questionnaire (Appendix G).

In this extrinsic evaluation phase, we used the TAM3 extension introduced in Section 3.5. By evaluating EOU and QUAL on a task-specific basis, we accounted for the varying levels of artifact utilisation across different use cases. The three tasks evaluated by the participants were:

1. **Lifecycle Management:** This task evaluates how easily and effectively a user can introduce a new data contract or modify an existing one within the KG framework. Given the framework's DP to support iterative knowledge evolution (DP4; Section 4.5.2), this task will assess the quality and ease of use of the system's updates to the graph's nodes, edges, and constraints without requiring a rebuild.

2. **Knowledge Retrieval and Querying:** The second task assesses the extensibility of the Knowledge Graph for downstream information retrieval and Retrieval-Augmented Generation (RAG) workflows (NFR2, Section 4.5.1.2). Participants will evaluate the framework based on its ability to accurately answer specific questions (such as the CQs) regarding data product lineage, ownership, architecture, and governance constraints.

This will determine whether the structured metadata provides reliable insights useful to both human users and autonomous AI agents navigating the KG.

3. **Executable Policy Generation:** By evaluating the system's ability to act as a foundation for downstream governance automation (FR2, Section 4.5.1.1), we assess whether the artifact bridges the gap between semantic business agreements and machine-enforceable rules. Participants will evaluate the generated executable policy code, such as Open Policy Agent (OPA) or Rego rules, to ultimately assess its quality and usefulness.

For each task, we define the EOU and QUAL variables. These constructs have the same causal positions as their general counterparts from the proposed original model by Davis, and the task-specific instantiation is shown in Figure 6.4.

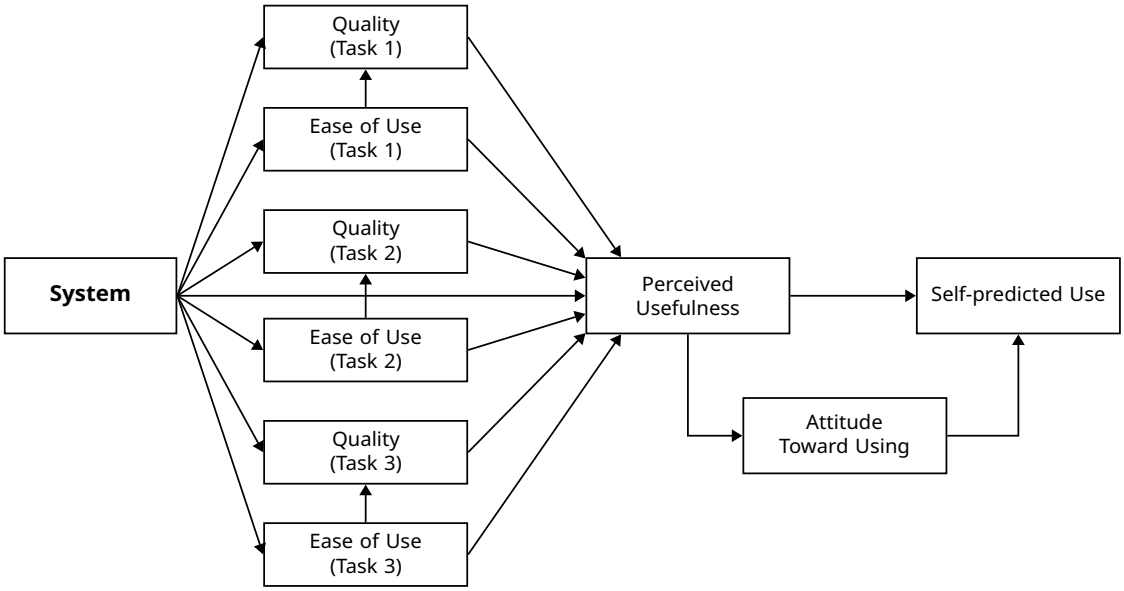

Figure 6.4: TAM adapted for evaluating the tasks [23]

The questions used to evaluate each construct are based on the polished 6-question list from Davis's original proposed TAM model [23]. The responses to these questions are recorded on a five-point Likert scale, where 1 denotes *Strongly Agree* and 5 denotes *Strongly Disagree*. This means that lower scores, therefore, indicate more favorable assessments. The internal consistency of each question list is assessed using Cronbach's alpha. Given the small response size, the results are interpreted descriptively as indicative evidence of acceptance rather than as a statistical test of the full TAM3 model.

Chapter 7

Results

7.1 Ontology Evaluation Results

7.1.1 Competency Questions

The ontology is evaluated by asking whether each competency question (CQ) can be expressed as a SPARQL query and answered against the gold standard graph; a CQ that cannot be expressed, or that returns no answer, points to a gap in the ontology rather than in the extraction framework (Section 6). To keep the evaluation close to practice, the CQs are presented through three short scenarios, each following one role around the *Customer Lifetime Value* data product: its owner, a governance officer, and a consumer. The scenarios are illustrative; what is actually being tested is whether every CQ returns a correct answer over the graph.

> **A note on node labels.** Throughout the figures, resource nodes follow the naming convention "<DataProduct><Node Type>", for example, the output port, input port, and a dataset of the Customer Lifetime Value product are named `CustomerLifetimeValueOutputPort`, `CustomerLifetimeValueInputPort`, and `CustomerLifetimeValueDataset` respectively. Because the visualisation truncates long captions, several distinct nodes may appear with the same leading text (e.g. `CustomerLifeti...`); they are nevertheless separate nodes, distinguished by the entity suffix of their full name. The same applies to the products in the dependency graphs, whose port and dataset nodes share the product-name prefix.

7.1.1.1 Newly Assigned Data Product Owner

A new owner takes over the Customer Lifetime Value product without prior context and must understand it before making any changes. They begin with the product's identity: its unique identifier (CQ1), name (CQ2), business domain (CQ3), description (CQ4), and intended use cases (CQ5), answered by the queries in Table 7.1.

The new owner starts with the most basic question: what is the data product? The result is shown in Figure 7.1. From these answers, the owner can already establish the product's scope: it is an aggregated historical view rather than a predictive model.

The owner also needs to know who is accountable, which team (CQ6), if the new owner is already the current owner (CQ7), the lineage of ownership (CQ9), and when these ownerships started (CQ8). The queries are listed in Table 7.2, where CQ7 and CQ8 share a query, differing only in the projected variable.

It is important for the new owner to know who held responsibility before and whether certain people have specific responsibilities within the data product. This information is all answered by the single ownership subgraph in Figure 7.2. The new owner can see that the previous owner was Sarah Connor and that the transition to the new owner is not in the system, while Alex Rivera is the primary contact for pipeline issues and SLA breaches, as he

CQ	Question	SPARQL query
CQ1	What is the unique identifier of the data product?	`SELECT ?product ?identifier WHERE { ?product a dprod:DataProduct ; dct:identifier ?identifier . FILTER(?identifier = "customer_lifetime_value") }`
CQ2	What is the name of the data product?	`SELECT ?name WHERE { ?product dct:identifier "customer_lifetime_value" ; rdfs:label ?name . }`
CQ3	Which business domain is associated with the data product?	`SELECT ?domain WHERE { ?product dct:identifier "customer_lifetime_value" ; dprod:domain ?domain . }`
CQ4	What is the business description of the data product?	`SELECT ?description WHERE { ?product dct:identifier "customer_lifetime_value" ; dct:description ?description . }`
CQ5	What is the intended use case for this data product?	`SELECT ?purpose ?usageNote WHERE { ?product dct:identifier "customer_lifetime_value" . OPTIONAL { ?product dprod:purpose ?purpose . } OPTIONAL { ?product skos:usageNote ?usageNote . } }`

Table 7.1: Competency questions CQ1–CQ5 mapped to SPARQL queries.

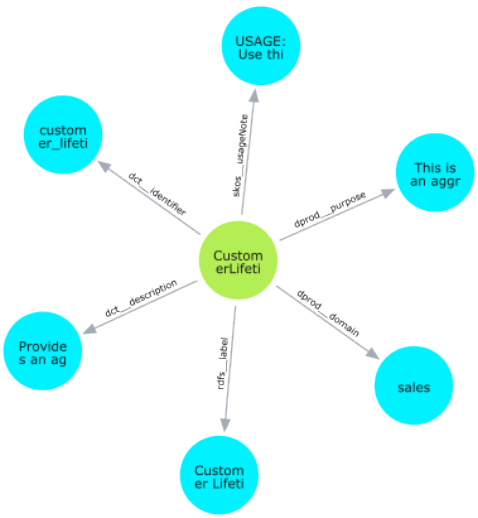

Figure 7.1: Identity of the Customer Lifetime Value data product, answering CQ1–CQ5: the product node linked to its identifier, name, domain, purpose, and usage-note literals.

CQ	Question	SPARQL query
CQ6	Which team is responsible for the data product?	`SELECT ?owner ?team WHERE { ?product dct:identifier "customer_lifetime_value" . ?ownership a dcont:ProductOwnership ; dcont:dataProduct ?product ; dprod:dataProductOwner ?owner . ?membership a org:Membership ; org:member ?owner ; org:organization ?team . }`
CQ7, CQ8	Who is the current data product owner? / When did the current data product owner start?	`SELECT ?owner ?startedAt WHERE { ?product dct:identifier "customer_lifetime_value" . ?ownership a dcont:ProductOwnership ; dcont:dataProduct ?product ; dprod:dataProductOwner ?owner ; prov:startedAtTime ?startedAt . } ORDER BY DESC(?startedAt) LIMIT 1`
CQ9	What is the historical timeline of ownership?	`SELECT ?owner ?startedAt ?note WHERE { ?product dct:identifier "customer_lifetime_value" . ?ownership a dcont:ProductOwnership ; dcont:dataProduct ?product ; dprod:dataProductOwner ?owner ; prov:startedAtTime ?startedAt . OPTIONAL { ?ownership dct:description ?note . } } ORDER BY ASC(?startedAt)`

Table 7.2: Competency questions CQ6–CQ9 mapped to SPARQL queries.

is the data engineer. So when something about the data product is unclear, it is best to ask Alex first before contacting Sarah.

The new owner also needs to know how the data product fits in the data architecture and what its dependencies are. Questions include which dataset it uses (CQ10), which output technology exposes the data product (CQ11), and the upstream (CQ12) and downstream dependencies (CQ13). The queries are listed in Table 7.3.

CQ	Question	SPARQL query
CQ10	Which dataset is used for the data product?	`SELECT ?ds WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputDataset ?ds . }`
CQ11	Which output port technologies expose the data product?	`SELECT ?protocol ?url WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . OPTIONAL { ?port dprod:protocol ?protocol . } OPTIONAL { ?port dcat:endpointURL ?url . } }`
CQ12	What are the data product's upstream dependencies?	`SELECT DISTINCT ?up WHERE { ?t dct:identifier "customer_lifetime_value" ; dprod:inputPort ?in . ?in dct:source ?upOut . ?up dprod:outputPort ?upOut . }`
CQ13	What are the data product's downstream dependencies?	`SELECT DISTINCT ?down WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?out . ?dIn dct:source ?out . ?down dprod:inputPort ?dIn . }`

Table 7.3: Competency questions CQ10–CQ13 mapped to SPARQL queries.

The dependency subgraph is shown in Figure 7.3. This shows the new data product owner how it is connected and what might break if a specific part is changed. The downstream products are most important, as the teams handling them may encounter breaking changes when the Customer Lifetime Value data product changes. The owners of these data products are added to the notify list when major version changes occur.

Finally, the new owner inspects the data the product guarantees: the dataset schema (CQ14), the metadata describing it (CQ15), and which fields are required (CQ16) or sensitive (CQ17). The queries are listed in Table 7.4

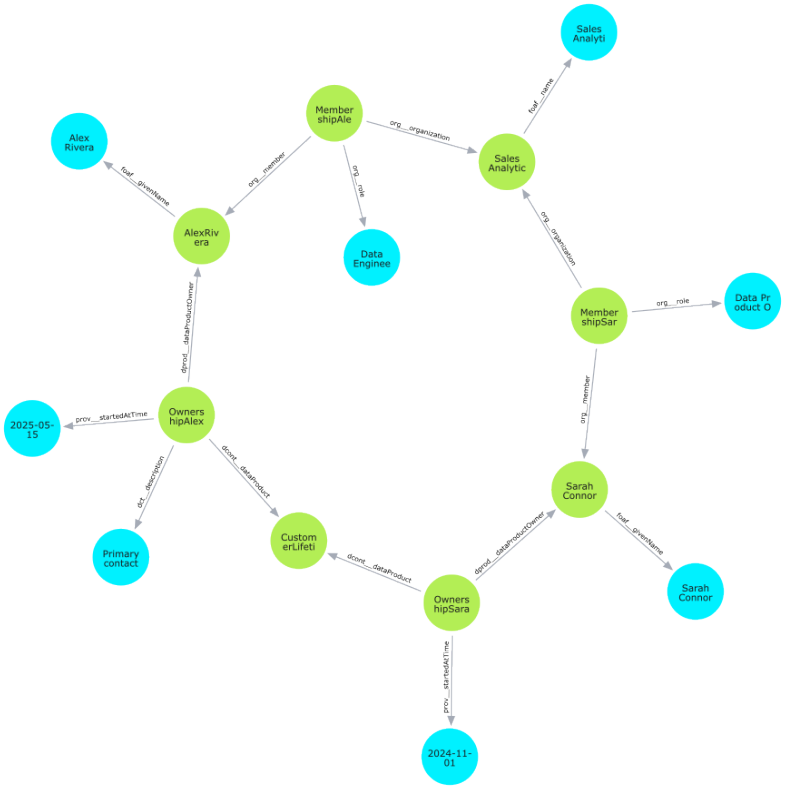

Figure 7.2: Ownership subgraph answering CQ6–CQ9: both ownership records with their start-date literals, the owners, their memberships, and the owning team.

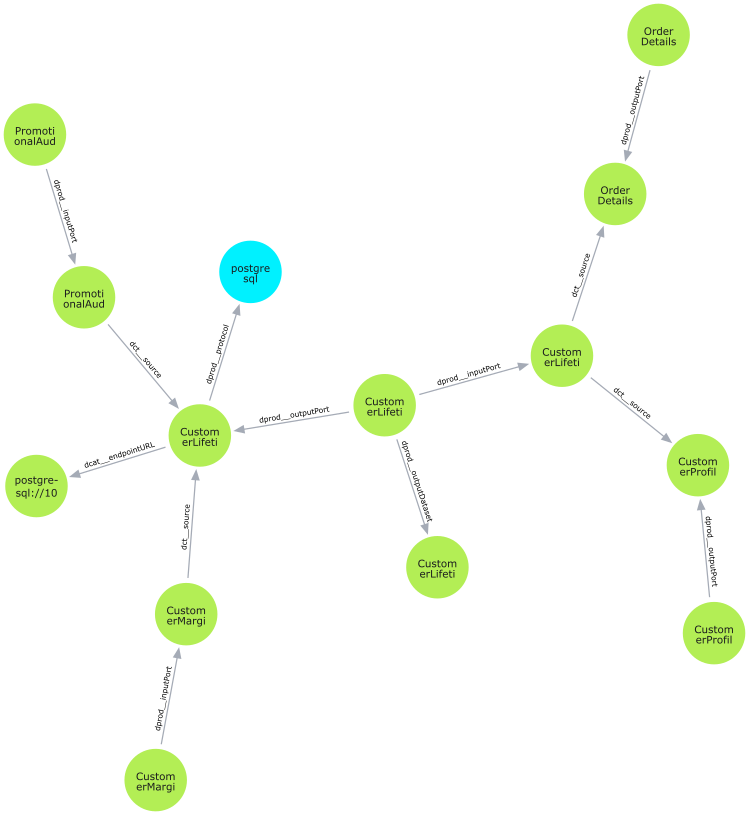

Figure 7.3: Dataset, output port, and upstream/downstream dependencies of the Customer Lifetime Value data product, answering CQ10–CQ13.

CQ	Question	SPARQL query
CQ14	What is the dataset schema?	`SELECT ?schema ?pk WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputDataset/dct:conformsTo ?schema . OPTIONAL { ?schema csvw:primaryKey ?pk . } }`
CQ15	What metadata describes the schema?	`SELECT ?col ?name ?datatype WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputDataset/dct:conformsTo ?schema . ?schema csvw:column ?col . ?col csvw:name ?name . OPTIONAL { ?col csvw:datatype ?datatype . } }`
CQ16	Which fields are required in the dataset?	`SELECT ?name WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputDataset/dct:conformsTo ?schema . ?schema csvw:column ?col . ?col csvw:required true ; csvw:name ?name . }`
CQ17	Which fields are sensitive in the dataset?	`SELECT ?name ?theme WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputDataset/dct:conformsTo ?schema . ?schema csvw:column ?col . ?col dcat:theme ?theme ; csvw:name ?name . FILTER(?theme IN (dpv:PersonalData, dpv:SensitivePersonalData)) }`

Table 7.4: Competency questions CQ14–CQ17 mapped to SPARQL queries.

After inspecting the data product guarantees from the schema subgraph shown in Figure 7.4, it is clear what the data schema looks like and which PII fields are present.

Within an hour, the owner has reconstructed the product's scope, ownership, dependencies, and schema from the graph alone, without a handover meeting.

7.1.1.2 Governance Officer Reviewing the Data Contract

When the Customer Lifetime Value pipeline fails to refresh overnight, the data becomes stale and may breach the contract. A governance officer then has to establish whether an obligation was actually broken, who is accountable, and which downstream consumers are affected, all of which the data contract is expected to answer. The officer first retrieves the contract's identifier (CQ18), its SLAs (CQ19), SLOs (CQ20), physical location (CQ21), the products it applies to (CQ22), and the parties involved (CQ23), using the queries in Table 7.5.

The first task is to establish what was contractually promised. The contract subgraph is shown in Figure 7.5, answering the above CQs. The relevant finding for this incident is the freshness obligation, currentness within 24 hours, which is exactly the guarantee that the missed refreshment has violated. The time-to-repair obligation establishes the remediation deadline, while the parties' nodes identify which parties use this data product. Finally, the authoritative contract document (CQ21) can also be located; it is an Excel document in an S3 bucket.

With the breach confirmed, the remaining task is to determine which data products and which consumers are affected (CQ24), so that an accurate impact notification can be sent. The mapped SPARQL query is seen in Table 7.6.

The impact subgraph is shown in Figure 7.6. The stale data is consumed by two downstream products and three assigned consumers, who will receive a notification about it.

From the contract alone, the officer can confirm the breach of obligation, identify the responsible parties, and locate the affected downstream consumers without consulting the owning team.

7.1.1.3 Consumer Evaluating Access and Obligations

A marketing analyst wants to use the Customer Lifetime Value product for a campaign and, before requesting access, needs to know what is permitted, whether their team already has access, and what obligations come with use. They ask which actions are prohibited (CQ25),

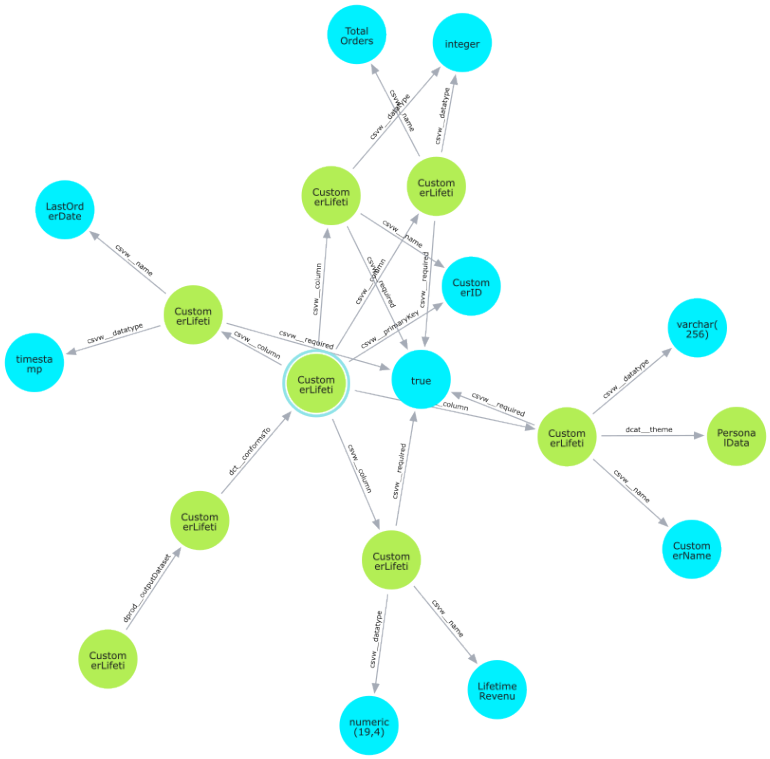

Figure 7.4: Schema subgraph answering CQ14–CQ17: the schema, its five columns with name/datatype literals, the required flags, and the dpv:PersonalData theme on CustomerName.

CQ	Question	SPARQL query
CQ18	What is the unique identifier of the data contract?	`SELECT ?cid WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port ; dct:identifier ?cid . }`
CQ19, CQ20	What are the service level agreements of the data contract? / What are the service level objectives of the data contract?	`SELECT ?metric ?op ?value ?unit WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port ; odrl:obligation ?d . ?d odrl:constraint ?k . ?k odrl:leftOperand ?metric ; odrl:operator ?op ; odrl:rightOperand ?value . OPTIONAL { ?k odrl:unit ?unit . } }`
CQ21	Where is the data contract located?	`SELECT ?loc WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port . OPTIONAL { ?c prov:hadPrimarySource ?loc . } }`
CQ22	To what data products does a data contract apply?	`SELECT DISTINCT ?app WHERE { ?c dct:identifier "CustomerLifetimeValue_contract" ; odrl:target ?port . ?app dprod:outputPort ?port . }`
CQ23	Who are the parties involved in a data contract?	`SELECT ?role ?party WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port . { ?c odrl:assigner ?party . BIND("assigner" AS ?role) } UNION { ?c odrl:assignee ?party . BIND("assignee" AS ?role) } }`

Table 7.5: Competency questions CQ18–CQ23 mapped to SPARQL queries.

CQ	Question	SPARQL query
CQ24	Which data products and consumers are impacted when a data contract is breached?	`SELECT DISTINCT ?type ?impacted WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port . { ?dIn dct:source ?port . ?impacted dprod:inputPort ?dIn . BIND("downstream" AS ?type) } UNION { ?c odrl:assignee ?impacted . BIND("consumer" AS ?type) } }`

Table 7.6: Competency question CQ24 mapped to its SPARQL query.

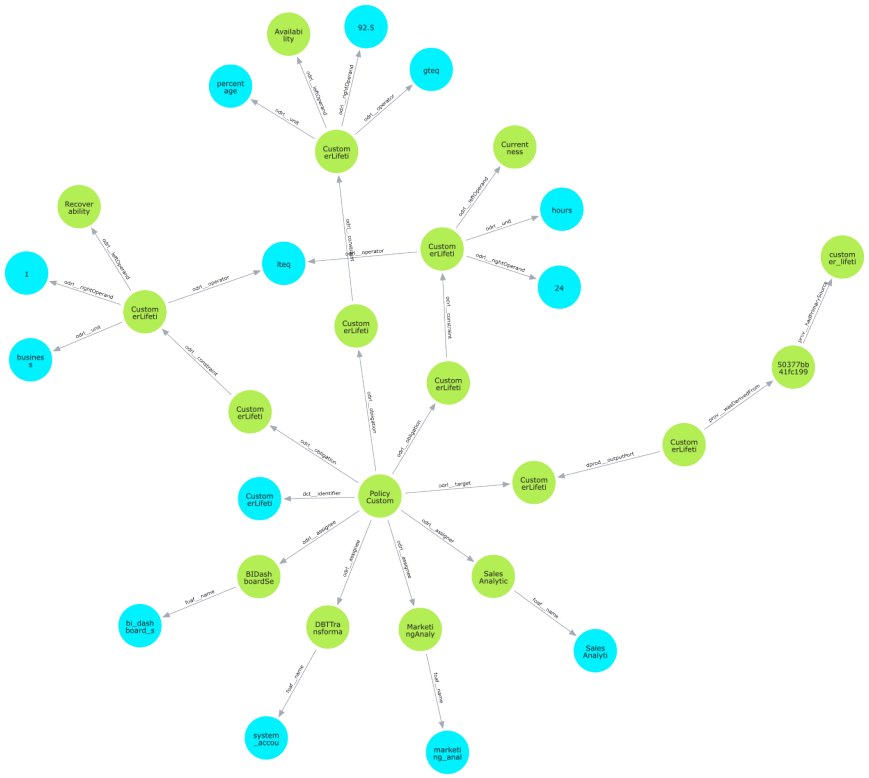

Figure 7.5: Data-contract subgraph answering CQ18–CQ23: the contract, its obligations and constraints with quality-metric and value literals, and the assigner/assignee parties.

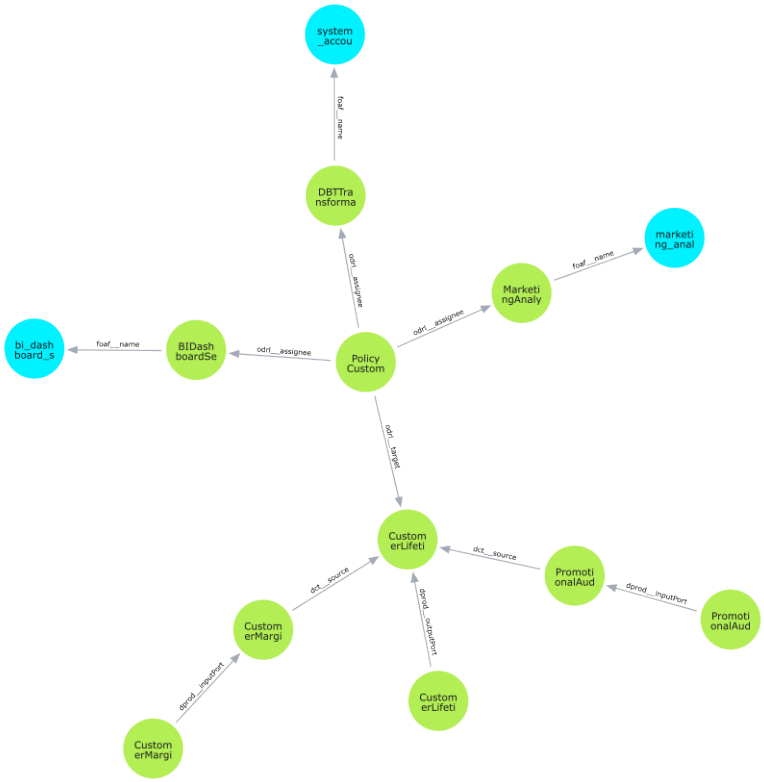

Figure 7.6: Products and consumers impacted by a breach of the data contract (answer to CQ24).

the expected usage (CQ26), who already has access (CQ27), which individuals hold approved access (CQ28), and what obligations apply (CQ29), via the queries in Table 7.7.

CQ	Question	SPARQL query
CQ25	Which actions are prohibited for consumers?	`SELECT ?action ?desc WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port ; odrl:prohibition ?pr . OPTIONAL { ?pr odrl:action ?action . } OPTIONAL { ?pr dct:description ?desc . } }`
CQ26	What is the expected usage of this data product?	`SELECT ?detail WHERE { ?p dct:identifier "customer_lifetime_value" . { ?p dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port ; odrl:permission ?perm . ?perm dct:description ?detail . } UNION { ?p skos:usageNote ?detail . } }`
CQ27	Who have access to this data product?	`SELECT DISTINCT ?party ?action WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port ; odrl:permission ?perm . ?perm odrl:assignee ?party . OPTIONAL { ?perm odrl:action ?action . } }`
CQ28	Which people have approved access?	`SELECT DISTINCT ?team ?person WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port ; odrl:assignee ?team . ?m a org:Membership ; org:organization ?team ; org:member ?person . }`
CQ29	What obligations must a consumer agree to?	`SELECT ?party ?action ?desc WHERE { ?p dct:identifier "customer_lifetime_value" ; dprod:outputPort ?port . ?c a dcont:DataContractPolicy ; odrl:target ?port ; odrl:permission ?perm . ?perm odrl:assignee ?party ; odrl:action ?action . OPTIONAL { ?perm dct:description ?desc . } }`

Table 7.7: Competency questions CQ25–CQ29 mapped to SPARQL queries.

The governance subgraph is shown in Figure 7.7. The results show the market analyst that their team already holds masked read access, so no new access request is required. It is not allowed to distribute the data to external parties. With that information, the marketing team can design a new campaign by joining this data with other data on the CustomerID, as they cannot read the CustomerName field (PII).

From the contract alone, the consumer can establish their access rights and obligations, and the same queries surface the terms they must accept if further access is needed, without involving the owning or governance teams.

Across all three roles, every competency question could be expressed as a SPARQL query and returned a correct, non-empty answer over the gold-standard graph. No CQ exposed a missing class, property, or relationship, which indicates that the Data Contract Ontology is complete with respect to the information needs identified in Section 5.2.1.

7.1.2 Ontology Pitfall Scanner (OOPS!)

Initially, the relationship mapping a data product to its owner (`dprod:dataProductOwner`) and the relationship mapping an owner to a data product (`dcont:dataProduct`) were considered an `owl:inverseOf` axiom. However, to accurately capture the lineage of ownership of a data product, an intermediary association class (`dcont:ProductOwnership`) was introduced. Because this pattern shifts the `dprod:dataProductOwner`'s domain of the relationship away from `prov:Agent` and onto the ownership entity itself, the OWL inverse caused the OOPS! scanner to flag a critical pitfall (*P5:* Defining wrong inverse relationship). After this pitfall was detected, we omitted this OWL inverse axiom to preserve the consistency of the model.

After this correction, the scan reports no critical pitfalls (Appendix F). The highest remaining severity is *important* (P11, missing domain or range), and every remaining flag concerns

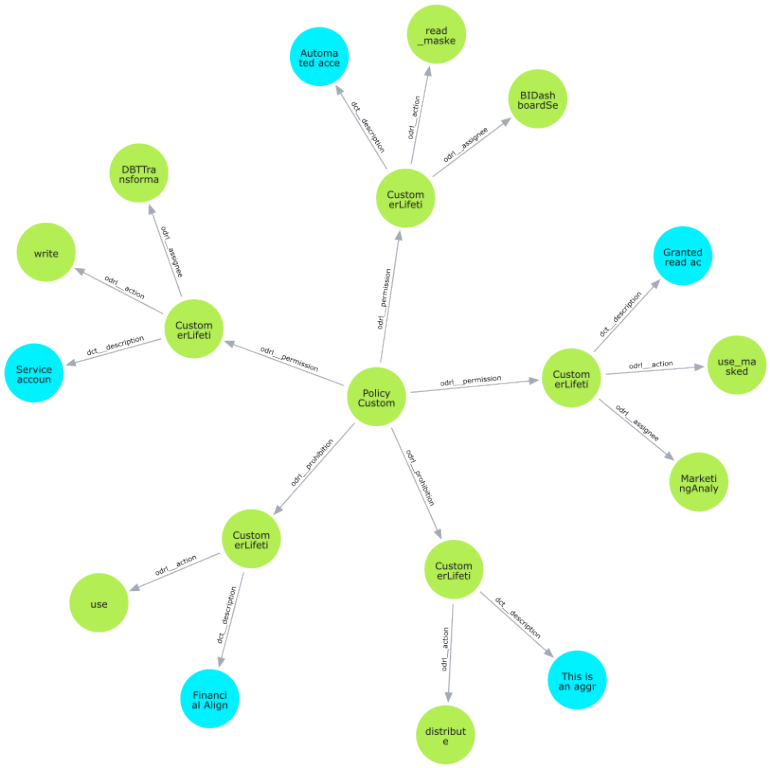

Figure 7.7: Governance subgraph answering CQ25–CQ29: prohibitions with their actions, permissions with their assigned parties and actions, and the individual members holding approved access.

an imported vocabulary, with one exception: `dcont:dataProduct` is now flagged under the minor pitfall *P13: inverse not explicitly declared*, which is the expected trade-off of removing the inverse axiom. The pitfalls of imported vocabulary fall outside the scope of this research and were left unmodified.

7.2 Intrinsic Evaluation Results

This section presents the results of the intrinsic evaluation of the extraction approach, assessing how accurately and efficiently the proposed pipeline retrieves the target triples from contracts under the different prompting & extraction strategies and contract formats considered in this research.

The evaluation is organized into two parts. Section 7.2.1 reports extraction performance across prompting & extraction strategies and contract formats, combining accuracy measures (F1-score, exact triple matches) with efficiency indicators (token usage, execution time) and how deterministic the answers are. It closes with an error analysis (Section 7.2.1.3) that breaks down failures by extraction stage and examines the violations identified during SHACL validation. Section 7.2.2 then turns to the consistency of the model's outputs, examining the entropy ratio of the main chain in comparison to the whole graph.

7.2.1 Extraction Evaluation

7.2.1.1 Metrics

Table 7.8 reports precision, recall, and F1-score for every configuration. The configurations fall into two non-overlapping groups according to whether the prompt contains an example: the weakest example-based configuration (one-shot and OS-CoT) still outperforms the strongest example-free (zero-shot and standard CoT) one, in both extraction modes and across all three contract formats. The example-free prompts stay uniformly low throughout. The separation is most pronounced for text-based contracts in direct mode, where the example-free F1 (0.143) is roughly five times lower than the example-based scores (0.722). The gap between the prompt strategies shows that giving an example carries more weight than an instruction to reason.

Two factors explain most of the variation in Table 7.8: whether the prompt contains an example, and the extraction mode. Example-based prompts dominate (highest F1-score in direct mode: 0.781 for one-shot and 0.773 for OS-CoT, versus 0.190 for zero-shot and 0.177 for CoT). Within the example-based prompts, the choice of contract format is less significant: YAML leads only marginally (one-shot direct 0.781 versus Excel with 0.736 and text with 0.722). The more consequential factor is the extraction mode: one-shot loses coherence when the task is split (YAML $0.781 \rightarrow 0.554$), whereas OS-CoT is largely consistent across modes (YAML $0.773 \rightarrow 0.730$).

Examining precision and recall separately shows that the two move closely together across all configurations. While precision is higher for the stronger configurations, it is only marginally (one-shot YAML direct: precision 0.812, recall 0.754, a gap of 0.058; OS-CoT YAML direct: precision 0.799, recall 0.749, a gap of 0.050). This indicates that when an example is shown, the framework produces slightly fewer incorrect triples than missing correct ones. The difference between precision and recall is sufficiently consistent that the F1-score balances both metrics well, justifying reliance on F1 as the primary metric for the remainder of the analysis. Without an example in the prompt, the two metrics remain nearly identical, indicating that the failure to extract the correct triples is not skewed towards any particular set.

Figure 7.8 shows the distribution of F1-scores across the ten runs per configuration, split by extraction mode, and the full per-configuration standard deviations are reported in Appendix H.1. The distribution confirms that the averages in Table 7.8 are quite accurate

	Direct Mode			Split Mode		
	YAML	*Excel*	*Text*	*YAML*	*Excel*	*Text*
Zero-shot						
Precision	0.187	0.165	0.139	0.205	0.226	**0.243**
Recall	0.194	0.169	0.148	0.203	0.220	**0.237**
F1	0.190	0.167	0.143	0.202	0.222	**0.239**
One-shot						
Precision	***0.812***	*0.777*	*0.762*	0.585	0.644	0.645
Recall	**0.754**	0.704	0.688	0.527	0.574	0.569
F1	**0.781**	0.736	0.722	0.554	0.606	0.603
CoT						
Precision	0.175	0.160	0.140	0.205	**0.253**	0.246
Recall	0.180	0.170	0.149	0.202	0.249	**0.249**
F1	0.177	0.164	0.144	0.203	**0.250**	0.247
OS-CoT						
Precision	**0.799**	0.771	0.749	*0.768*	*0.753*	*0.733*
Recall	**0.749**	0.702	0.693	0.695	0.665	0.629
F1	**0.773**	0.735	0.719	0.730	0.705	0.676

Table 7.8: Precision, Recall, and F1 by prompting strategy, contract format, and mode. For each row, the best value is shown in **bold** and for each configuration, the best value is shown in *italic.*

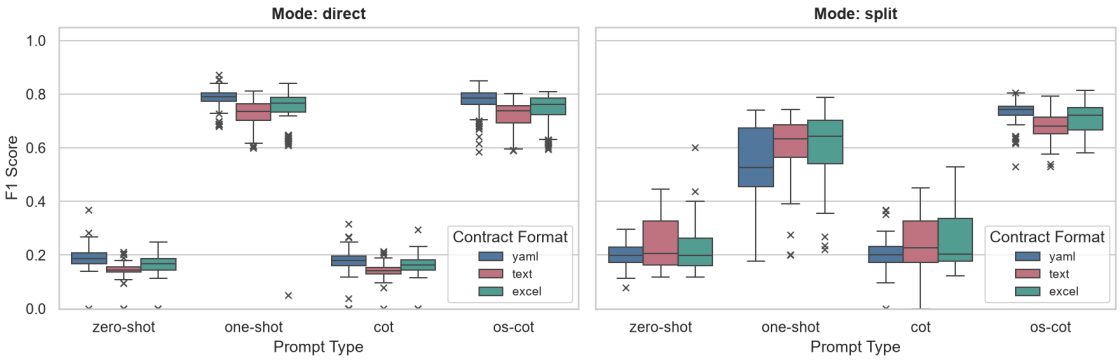

Figure 7.8: Distribution of F1-score by prompting strategy and contract format

and that these means do not conceal high variance for most configurations. The spread is widest for one-shot in split mode (std 0.133 for YAML) and narrowest for CoT in direct mode (std 0.028 for text), although this narrow spread reflects consistently poor extraction rather than reliable performance.

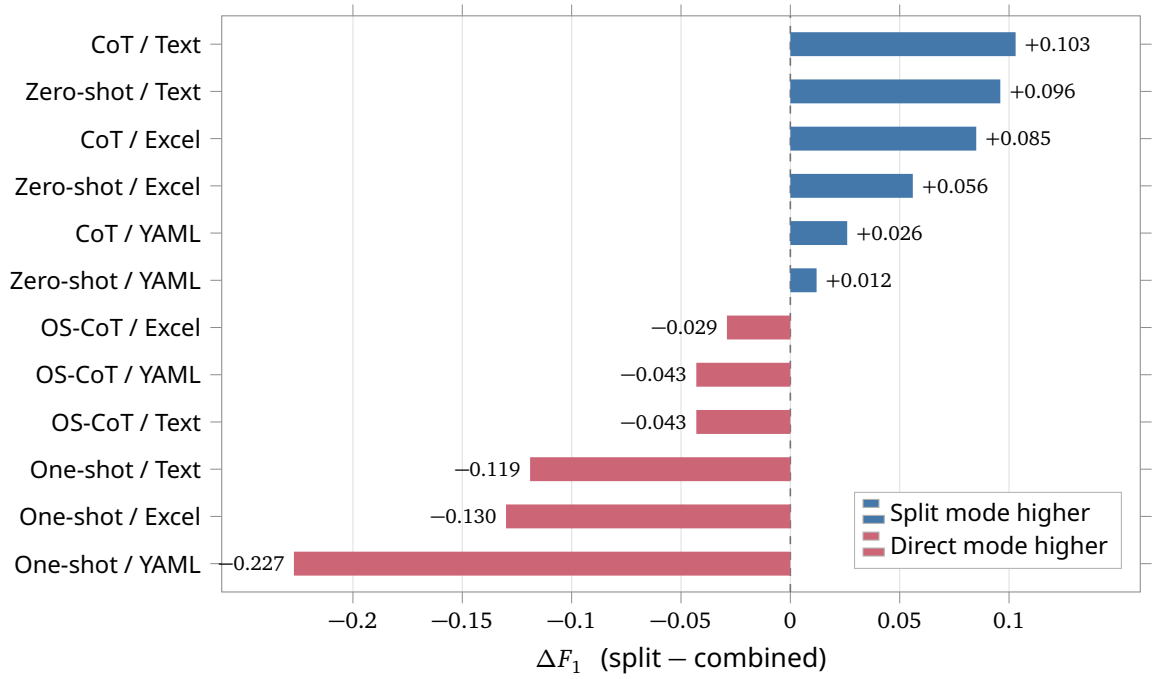

Figure 7.9: Difference in F_1 between split and direct modes for each prompting strategy and contract format, $\Delta F_1 = F_1^{\text{split}} - F_1^{\text{direct}}$. Positive values indicate higher performance under split mode

Figure 7.9 summarises the difference between direct and split modes. The effect of splitting is not uniform; it raises the prompts without an example but lowers the performance of example-based prompts. The gains for the weak strategies are small (at most $+0.103$), and they increase performance only from around 0.14 to 0.25. The decrease in performance for the example-based prompts is substantial, and most affects the best configuration (the direct YAML one-shot drops by 0.227, from 0.781 to 0.554).

In Table 7.9, performance is shown by the raw number of generated triples and their share of matches with the gold standard. All strategies generate a comparable number of triples (227 ± 15), so the difference in F1-score is driven mostly by match quality rather than quantity. The exact-match share is far higher for example-based prompts than for example-free ones (YAML direct: 75.1% for one-shot vs. 37.4% for zero-shot). The split-mode column shows where one-shot loses ground: its exact-match count collapses, whereas OS-CoT's stays comparable. The drop is therefore a real loss of correct triples, not merely a lower F1.

The fuzzy matches in Table 7.9b require a more careful reading, because these are not exact matches, and having a high amount of fuzzy matches can be a bad sign. For weaker prompt strategies without an example, the fuzzy share of matches is high (YAML split CoT reaching 80.3%), but this means the output with this configuration is often a near miss relative to the gold-standard triple. For stronger prompt strategies, the number of fuzzy matches is lower; this is because there are more exact matches, and therefore fewer triples to capture during fuzzy match-making.

Table 7.10 reports the average token consumption per extraction loop. It is interesting to note that the main driver of increased token usage is not the prompting strategy or contract format, but the extraction mode. Split mode consumes roughly two to three times

	Direct Mode			Split Mode		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
Generated	254.6	236.8	241.2	248.7	234.5	227.4
Matches	17.9	19.2	**19.0**	15.2	18.6	18.8
Share (%)	37.4	48.3	**55.6**	30.3	35.8	34.4
One-shot						
Generated	228.5	212.5	209.5	221.6	208.9	204.8
Matches	**139.3**	125.5	117.9	88.8	96.4	93.9
Share (%)	75.1	**75.8**	73.9	68.3	71.7	71.4
CoT						
Generated	241.5	244.8	244.0	243.0	232.5	230.3
Matches	15.5	**18.9**	18.7	9.8	18.5	16.2
Share (%)	35.0	47.3	**54.4**	19.7	31.7	28.3
OS-CoT						
Generated	230.5	214.0	215.0	222.1	207.4	198.9
Matches	**139.4**	123.7	118.9	119.9	112.3	106.9
Share (%)	**75.7**	75.1	74.0	70.3	72.0	73.4

(a) Exact matches

	Direct Mode			Split Mode		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
Generated	254.6	236.8	241.2	248.7	234.5	227.4
Matches	30.0	20.5	15.1	34.9	33.3	**35.9**
Share (%)	62.6	51.7	44.4	**69.7**	64.2	65.6
One-shot						
Generated	228.5	212.5	209.5	221.6	208.9	204.8
Matches	**46.2**	40.0	41.7	41.3	38.1	37.6
Share (%)	24.9	24.2	26.1	**31.7**	28.3	28.6
CoT						
Generated	241.5	244.8	244.0	243.0	232.5	230.3
Matches	28.8	21.1	15.7	40.1	39.9	**41.0**
Share (%)	65.0	52.7	45.6	**80.3**	68.3	71.7
OS-CoT						
Generated	230.5	214.0	215.0	222.1	207.4	198.9
Matches	44.8	41.1	41.8	**50.7**	43.8	38.7
Share (%)	24.3	24.9	26.0	**29.7**	28.0	26.6

(b) Fuzzy matches

Table 7.9: Matches by prompting strategy, contract format, and mode. Share (%) is the percentage of matched triples that are exact (table a) versus fuzzy (table b); the exact and fuzzy shares for a configuration therefore sum to 100%

	Direct Mode			Split Mode		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
Input tokens	49,784	46,782	**46,243**	119,076	118,658	110,399
Output tokens	21,083	20,137	**18,647**	46,425	46,904	43,728
Total tokens	70,931	65,732	**65,378**	165,501	165,562	154,127
One-shot						
Input tokens	23,657	23,047	**20,678**	81,324	67,347	60,860
Output tokens	10,157	10,102	**8,628**	28,041	24,808	21,166
Total tokens	33,059	32,600	**29,334**	109,365	92,155	82,027
CoT						
Input tokens	51,664	43,639	**39,134**	133,740	117,414	123,558
Output tokens	23,071	20,092	**17,059**	55,498	48,367	51,344
Total tokens	74,778	63,148	**57,150**	189,238	165,780	174,902
OS-CoT						
Input tokens	17,913	**15,782**	19,208	68,296	52,543	49,529
Output tokens	5,797	**5,020**	5,888	25,831	19,493	16,789
Total tokens	23,710	**20,802**	25,096	94,128	72,036	66,318

Table 7.10: Average token usage by prompting strategy, contract format, and mode

as many total tokens as its direct-mode counterpart across all configurations. This makes sense, as for each action in the split extraction pipeline, entities are extracted first, and then, using those entities, relations are extracted, which means submitting the prompt with the contract context twice. Among all configurations, the OS-CoT is the most efficient, using only $23,710$ tokens for direct YAML extraction and $20,802$ tokens for Excel contract formats. The most expensive strategies are the CoT and zero-shot split extractions, as, besides the split extraction pipeline, these prompt strategies have to re-run more often due to errors in the syntax or the SHACL shapes (discussed later in Section 7.2.1.3).

Execution time, reported in Appendix H.3, follows the same pattern as token consumption: direct mode is consistently faster than split mode, and OS-CoT in direct mode is the fastest configuration overall, indicating that the cheapest strategy in tokens is also the quickest to run.

7.2.1.2 Deterministic Analysis

Because the extraction pipeline relies on a non-deterministic LLM, identical output is not guaranteed. To quantify this, the resulting triples are compared, and stability is measured by the CV of the number of generated triples and by the Jaccard Index for the overlap of triple sets across runs. Table 7.11 reports both measurements, with the Jaccard Index given in exact and fuzzy matches.

	Direct Mode			**Split Mode**		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
CV Triples gen.	0.091	0.135	0.121	0.126	0.107	**0.066**
Jaccard exact	0.255	0.301	**0.351**	0.234	0.272	0.281
Jaccard fuzzy	0.871	0.849	0.877	**0.913**	0.877	0.876
One-shot						
CV Triples gen.	0.044	0.067	**0.027**	0.038	0.050	0.060
Jaccard exact	0.786	**0.793**	0.774	0.505	0.584	0.557
Jaccard fuzzy	**0.968**	0.962	0.950	0.957	0.943	0.925
CoT						
CV Triples gen.	0.190	0.115	**0.093**	0.123	0.100	0.150
Jaccard exact	0.273	0.354	**0.410**	0.294	0.262	0.193
Jaccard fuzzy	0.881	0.897	**0.906**	0.872	0.873	0.839
OS-CoT						
CV Triples gen.	0.051	**0.029**	0.058	0.030	0.046	0.050
Jaccard exact	0.795	0.812	0.761	**0.815**	0.755	0.722
Jaccard fuzzy	0.969	0.967	0.951	**0.980**	0.966	0.962

Table 7.11: Extraction determinism: run-to-run coefficient of variation (CV), where lower values indicate greater stability, and content Jaccard, where higher values indicate greater stability.

As we already saw in Table 7.9, the volume of output triples is stable across all strategies. CV values are mostly between 0.03 and 0.15, with lower values indicating less variability. The example-based prompt strategies show the highest stability with direct text extraction, with one-shot prompting at 0.027. CoT is the most volatile when extracted directly from YAML, with a CV of 0.190. While the spread is narrow, triple count alone is not sufficient for

determinism, as it only determines the variability and does not say anything about which triples are produced.

The Jaccard Index does exactly what the triple counts can't measure. The Jaccard exact counts the number of triples when the normalized triples are identical across runs, while the Jaccard fuzzy counts a match when the triples match for more than 90%. The difference between configurations is quite high in terms of output determinism. The example-based strategies are highly reproducible for exact matches in direct mode (one-shot YAML 0.786 and OS-CoT YAML 0.795), whereas zero-shot and CoT decrease this stability to $0.25 - 0.41$, meaning that the majority of produced triples differ from run to run. The fuzzy Jaccard is high across configurations ($0.84 - 0.98$), indicating that, in most runs, the extracted triples are semantically similar but expressed with slight differences.

An important note is about the one-shot Jaccard Index difference between the extraction modes. For YAML extraction, the exact Jaccard measure drops from 0.786 to 0.505. This confirms that splitting destabilizes one-shot prompting, not only in performance, but also in deterministic output.

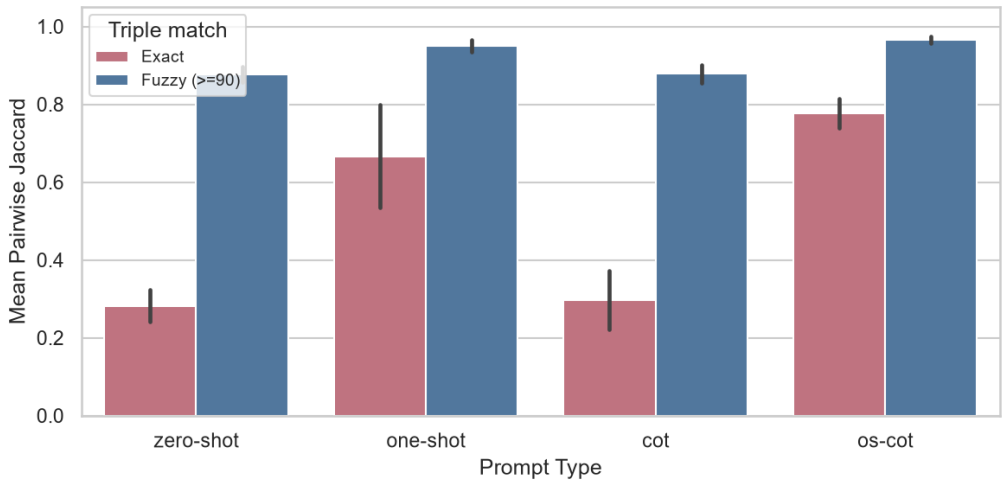

Figure 7.10: Mean pairwise Jaccard similarity of extracted triples by prompt strategy. The plot contrasts exact triple matching with fuzzy matching (similarity ≥ 0.90). Higher values indicate greater extraction determinism across multiple runs.

Figure 7.10 aggregates these results by strategy, making the pattern visible at a glance. The fuzzy bars are high for all strategies, while the exact Jaccard Index bars are split into two: one-shot and OS-CoT are high, and zero-shot and CoT are significantly lower. The error bars are also interesting: the one-shot prompting strategy shows a wide interval, reflecting underlying instability in split mode, whereas the OS-CoT shows a consistent Jaccard Index.

7.2.1.3 Error Analysis

To locate where the pipeline fails rather than only how often, each extraction outcome is classified as valid, SHACL violation, or syntax error, as reported in Table 7.12. The dominant failure mode is more semantic than syntactic: the pipeline almost always produces valid RDF triples, but these often still violate the ontology's structural constraints.

The split between strategies is high: example-based strategies pass SHACL validations in the majority of cases, with some configurations even producing zero SHACL errors (direct one-shot with text format and direct OS-CoT with Excel and text format). The prompt strategies without an example are the opposite: for example, zero-shot is only completely valid in

$6.7 - 20\%$ of cases, with the remainder often failing SHACL validation, while CoT performs slightly better, with a range of $37.8 - 57.8\%$ valid cases.

For most configurations, splitting the extraction mode decreases the validity percentage (except for zero-shot text). Especially for CoT strategies, the valid rate collapses from $37.8 - 57.8\%$ to $3.3 - 16.7\%$ in split mode. One-shot and OS-CoT degrade only slightly, which is consistent with the other performance metrics.

Beyond the count, it is worth characterizing the type of structural errors the different prompting strategies produce. Without an example, the framework produces many structural errors.

Grouping the violations by the ontology region they breach (Table 7.13) shows that failures are concentrated rather than spread out. The ODRL policy model accounts for roughly 45% of all 3,913 violations (Table 7.14) and about two-thirds of the most frequent violation types: obligations, their operands, and their operators. This is the most deeply nested and least lexically explicit part of a data contract, so the model must infer the policy structure rather than copy surface tokens, which is where example-free prompts fail most often. The second cluster is typing and datatype discipline (e.g. `csvw:datatype`, output-port class constraints), reflecting cases where the model produces structurally plausible but incorrectly typed nodes. The two clusters point to the same weakness: errors arise where correct output depends on the ontology's structural rules rather than on text that can be lifted directly from the contract.

Table 7.15 reports the average number of violations per extraction, separating syntax and SHACL errors, and refines the picture. A result that stands out is OS-CoT in direct extraction mode, which produces essentially no syntactic errors and the fewest SHACL violations of any configuration: its worst contract format still has a lower per-extraction violation rate than the best format of any other strategy. One-shot follows closely, while zero-shot and CoT produce several times more violations in both syntax and SHACL validation, and splitting the extraction increases errors in both error types across all configurations.

	Direct Mode			Split Mode		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
Valid count	14	14	7	6	13	18
SHACL errors	76	73	82	84	75	72
Syntax errors	**0**	3	1	**0**	2	**0**
Valid (%)	15.6	15.6	7.8	6.7	14.4	20.0
SHACL (%)	84.4	81.1	91.1	93.3	83.3	80.0
Syntax (%)	**0.0**	3.3	1.1	**0.0**	2.2	**0.0**
One-shot						
Valid count	86	87	90	81	83	80
SHACL errors	4	3	**0**	9	7	10
Syntax errors	**0**	**0**	**0**	**0**	**0**	**0**
Valid (%)	95.6	96.7	**100.0**	90.0	92.2	88.9
SHACL (%)	4.4	3.3	**0.0**	10.0	7.8	11.1
Syntax (%)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
CoT						
Valid count	34	50	52	3	12	15
SHACL errors	56	40	38	85	78	73
Syntax errors	**0**	**0**	**0**	2	**0**	2
Valid (%)	37.8	55.6	57.8	3.3	13.3	16.7
SHACL (%)	62.2	44.4	42.2	94.4	86.7	81.1
Syntax (%)	**0.0**	**0.0**	**0.0**	2.2	**0.0**	2.2
OS-CoT						
Valid count	89	90	90	84	86	82
SHACL errors	1	**0**	**0**	6	4	8
Syntax errors	**0**	**0**	**0**	**0**	**0**	**0**
Valid (%)	98.9	**100.0**	**100.0**	93.3	95.6	91.1
SHACL (%)	1.1	**0.0**	**0.0**	6.7	4.4	8.9
Syntax (%)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

Table 7.12: Extraction stage distribution by prompting strategy, contract format, and mode

Ontology region	Violated constraint	Total	Percentage
Policy / Governance			
`odrl:obligation`	not an IRI, not `odrl:Duty`, or fails duty shapes	**643**	**16.4%**
`odrl:leftOperand`	missing, repeated, or invalid type	374	9.6%
`odrl:operator`	missing, repeated, or not a literal	278	7.1%
output port	no policy targets this output port	265	6.7%
`odrl:rightOperand`	missing, repeated, or not a literal	209	5.3%
Typing / Schema			
`csvw:datatype`	missing, repeated, or invalid type	**470**	**12.0%**
`dprod:outputPort`	does not satisfy class/node constraints	321	8.2%
`dataProductOwner`	missing, not an IRI, or invalid type	147	3.7%

Table 7.13: Most frequent SHACL violation types, grouped by ontology region

	Direct Mode			Split Mode			
	YAML	Excel	Text	YAML	Excel	Text	**Total**
Zero-shot	252	211	238	559	359	231	**1850**
One-shot	4	4	**0**	17	38	42	105
CoT	172	90	89	490	538	527	**1906**
OS-CoT	2	**0**	**0**	7	14	29	**52**
Total	**430**	**305**	**327**	**1073**	**949**	**829**	**3913**

Table 7.14: Total SHACL violations by prompting strategy, contract format, and mode

	Direct Mode			Split Mode		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
Syntax errors	1.71	1.78	**1.39**	3.38	3.51	2.97
SHACL errors	2.76	2.76	2.86	2.91	2.88	**2.79**
One-shot						
Syntax errors	**1.03**	1.17	**1.03**	1.78	2.04	1.60
SHACL errors	0.78	0.87	**0.71**	1.86	1.40	1.20
CoT						
Syntax errors	2.40	2.11	**1.61**	3.91	3.21	3.57
SHACL errors	2.62	2.34	**2.16**	2.98	2.90	2.90
OS-CoT						
Syntax errors	**0.00**	0.01	0.01	1.90	1.62	**1.17**
SHACL errors	0.63	**0.54**	0.83	1.30	0.90	**0.91**

Table 7.15: Average violations per extraction across prompt strategies, contract formats, and mode

7.2.2 Extraction Entropy

Finally, the entropy ratio assesses how much of the generated graph's structural complexity is concentrated in the governance main chain, as defined in Section 6.3.1.3. A higher ratio means the core governance subgraph accounts for most of the graph's complexity, while a lower percentage means descriptive metadata dominates. Table 7.16 reports the mean entropy ratio per configuration with its standard deviation to show the distribution of the graph's main chain entropy across runs

	Direct Mode			Split Mode		
Strategy	**YAML**	**Excel**	**Text**	**YAML**	**Excel**	**Text**
Zero-shot	0.806 ± 0.030	0.805 ± 0.089	$\mathbf{0.813 \pm 0.089}$	0.788 ± 0.025	0.796 ± 0.020	0.805 ± 0.015
One-shot	0.822 ± 0.006	0.831 ± 0.008	$\boldsymbol{0.834 \pm 0.005}$	0.812 ± 0.007	0.821 ± 0.009	0.824 ± 0.008
CoT	0.798 ± 0.152	0.813 ± 0.124	$\mathbf{0.822 \pm 0.088}$	0.792 ± 0.087	0.808 ± 0.016	0.810 ± 0.025
OS-CoT	$*0.823 \pm 0.005*$	$*0.833 \pm 0.006*$	0.833 ± 0.009	$*0.821 \pm 0.006*$	$*0.830 \pm 0.007*$	$\boldsymbol{0.836 \pm 0.007}$

Table 7.16: Mean main-chain entropy ratio ± standard deviation by prompting strategy, contract format, and extraction mode (ten runs per configuration). The best value in each row is shown in **bold** and the best in each column in *italics*.

As shown in Table 7.16, the mean entropy ratio is high and stable across all configurations (0.788 to 0.836), indicating that most of the graph's complexity lies within the governance-relevant part of the ontology. However, the distribution of the entropy ratio across runs, also reported in the same table, provides a contrasting view of the stability of this mean. First, variability across configurations is substantial: one-shot and OS-CoT produce compact distributions with few outliers, whereas zero-shot and CoT exhibit significantly greater variability, with many outliers. These outliers indicate that, for some contracts, the graph's focus had drifted away from the governance core. Second, splitting the extraction mode slightly stabilizes the distribution. This is most visible for the prompts without examples, because far fewer extractions fail to recover a governance chain in split mode (one versus eight in direct mode), which removes the zero-ratio outliers that inflate the spread.

For nine extractions, the main chain ratio was 0.0; for the direct extraction, it was three times CoT YAML, twice CoT Excel, once CoT text, once zero-shot Excel, and once zero-shot text, while for the split extraction, it happened once for CoT YAML. A ratio of zero means that the framework was unable to extract a governance chain from the data contract. These failures dominate the standard deviation of the affected configurations, as a zero sits far below their otherwise high entropy ratio; each additional failure increases the standard deviation significantly. The direct CoT YAML configuration, with the most failures (three), has the highest standard deviation (0.152), 0.028 above the next-highest, direct CoT Excel (0.124, two chain failures). This is a large gap given that the example-based configuration, which never fails, stays below 0.01.

[74]

7.3 Extrinsic Evaluation Results

As detailed in Section 6.4.1, six of the eleven industry experts interviewed during the exploratory phase completed the questionnaire for the three evaluation tasks. Participants took an average of 56.16 minutes to complete the survey. The remainder of this section presents the evaluation results.

First, Section 7.3.1 discusses the participants' familiarity with the topic and their perceived need. Next, the reliability of the questionnaire will be discussed in Section 7.3.2. Section 7.3.3 details the task outcomes, participant attitudes, and perceived usefulness. Section 7.3.4 then

examines the anticipated impact on governance. Finally, Section 7.3.5 reviews the responses to the open-ended feedback questions.

7.3.1 Familiarity and Perceived Need

Before the task-based evaluation, the participants indicated their prior familiarity with the concept of an ontology layer and whether they expected such a layer to improve their daily work. Familiarity was mixed, but five of the six respondents rated themselves as very or moderately familiar, and only QP6 rated themselves as somewhat familiar. Regarding the perceived need, five of the six agreed that an ontology layer describing data products would significantly improve related daily activities; only QP6 disagreed.

The five respondents who agreed were asked which of their daily activities would benefit most, while QP6, having disagreed, was instead asked why such a layer might not be helpful. The responses from the respondents who agreed are summarized in Table 7.17, concentrated on data discovery as *discovering and querying data products* was the most frequently selected benefit (4 of 5), followed by *automating data quality, SLA, and compliance checks*, *tracing data lineage, provenance, and ownership*, and *managing the lifecycle of data products/contracts*, which each was selected by three of the five. The pattern is consistent with the task-level results in Section 7.3.3: the capability respondents expect to use most (discovery) is also the one they rated easiest and most useful, whereas *Translating legal/business constraints into executable policies*, named only once, is also the lowest-rated task. One respondent (QP5) used the open *Other* field to describe benefits around consuming data products, connecting different data products, demonstrating their value, and enforcing policies and compliance.

Activity	Count (of 5)
Discovering and querying data products	4
Automating data quality, SLA, and compliance checks	3
Tracing data lineage, provenance, and ownership	3
Managing the lifecycle of data products/contracts	3
Translating legal/business constraints into executable policies	1
Other	1

Table 7.17: Activities expected to benefit most from the ontology layer; select-all-that-apply item derived from Table G.10. Answered by the five respondents who agreed an ontology layer would be useful (QP6, who disagreed, was routed to the alternative branch).

The only participant who disagreed whether an ontology layer would improve their daily work was QP6, and they attributed their choice to three factors: *Organizational inertia*, *the overhead of defining semantics and breaking down tacit knowledge*, and *the perceived immaturity of current vendor tooling and data catalogs for supporting an ontology layer*.

7.3.2 Reliability of the TAM3 Constructs

Before the results of the survey, it is useful to assess the reliability of each question set, that is, whether the items intended to measure each TAM3 construct do so consistently. This internal consistency is summarised by Cronbach's alpha (α), a coefficient that is typically interpreted on a $0-1$ scale. A high α indicates that the items in the question set are coherent, whereas a low value suggests the items may be measuring different things. Do note that the sample size is quite small and therefore does not support full statistical reliability, but it does indicate how reliably each construct was measured and flags where individual items diverge.

Table 7.18 reports Cronbach's alpha per question set. Most item set reach conventionally acceptable values for a 3-item set ($\alpha \geq 0.8$) [99], with many question sets being between 0.85

Construct	Items (k)	Cronbach's α
Task 1: Lifecycle Management		
Ease of Use (EOU)	3	0.932
Perceived Output Quality (QUAL)	3	0.847
Importance (IMPORT)	2	0.980
Task 2: Knowledge Retrieval and Querying		
Ease of Use (EOU)	3	0.659
Perceived Output Quality (QUAL)	3	-0.706
Importance (IMPORT)	2	0.868
Task 3: Executable Policy Generation		
Ease of Use (EOU)	3	0.886
Perceived Output Quality (QUAL)	3	0.906
Importance (IMPORT)	2	0.921
Framework-Level TAM3 Constructs		
Perceived Usefulness (USEF)	3	0.876
Attitude Toward Using (ATT)	3	0.906
Self-Predicted Use	2	0.857

Table 7.18: Cronbach's alpha reliability coefficients for each TAM3 construct in the extrinsic artifact evaluation ($n = 6$)

and 0.98. Two results stand out: the first is task 2, Ease of Use, which, at 0.659, falls below the usual threshold. The second, and more striking, is the task 2 Perceived Output Quality, which returns a negative α of -0.706.

A negative Cronbach's alpha does not indicate negative reliability; it is a known artifact of the statistic that can occur when the items are not positively correlated with one another [99]. Due to the question set size (3) and the responses being similar with a small variance, it is possible to get a negative internal consistency value. This value should therefore be read as an artifact of the small sample size rather than as evidence that the question set is unreliable.

7.3.3 Task Evaluation and TAM3 Constructs

Having established the internal consistency of the constructs, the mean results from the evaluation survey are shown in Table 7.19 while the raw responses can be found in Appendix G.2. The task-specific constructs (Ease of Use, Perceived Output Quality, and Importance) capture how the framework was experienced for each individual task, while the framework-level constructs (Perceived Usefulness, Attitude Toward Using, and Self-Predicted Use) capture the respondents' overall disposition towards the LACE Framework after completing all the tasks.

Table 7.19 reports the mean scores per task and at the framework level. Across all three tasks, the scores indicate an overall positive reception of the LACE framework, though they differ slightly by task.

Across the three tasks, the ranking is consistent and interpretable rather than purely numerical: Task 2 (knowledge retrieval and querying) is rated as the easiest to use and highest in output quality, and it is also the task respondents rated most important (Table 7.19). Task 3 (executable policy generation) is consistently the least favourably rated on

Construct	EOU	QUAL	IMPORT
Task-Specific Constructs			
Task 1: Lifecycle Management	1.94	2.33	2.42
Task 2: Knowledge Retrieval and Querying	1.39	2.06	1.75
Task 3: Executable Policy Generation	2.22	2.72	2.83
Framework-Level Constructs			
Perceived Usefulness (USEF)		1.94	
Attitude Toward Using (ATT)		2.28	
Self-Predicted Use		3.17	

Table 7.19: Mean TAM3 construct scores for the extrinsic artifact evaluation ($n = 6$), averaged across items and respondents. Task-specific Ease of Use (EOU), Perceived Output Quality (QUAL), and task Importance (IMPORT) are reported per task; Perceived Usefulness (USEF), Attitude Toward Using (ATT), and Self-Predicted Use are reported at the framework level. Raw scale: 1 = Strongly Agree to 5 = Strongly Disagree, so lower values indicate more positive evaluations.

every construct. In other words, the framework is received most positively exactly where respondents see the greatest need. One caveat: for Task 2, the reliabilities for Ease of Use and Perceived Output Quality were low ($\alpha = 0.659$ and -0.706), so the corresponding means are less internally consistent and should be read with caution.

Task importance shows a different pattern as it measures how essential the task is to the respondent's role rather than how well the framework performs it. Knowledge retrieval was rated as the most important task with an average score of 1.75, followed by lifecycle management (2.42) and finally policy generation (2.83).

At the overall framework level, the generalized Perceived Usefulness of the LACE Framework is rated relatively high at 1.94, and the Attitude Toward using it is also favorable (2.28). The Self-Predicted Use is a bit of an outlier with a mean score of 3.17, which is the only construct in the entire evaluation to land more on the disagreeing side of the Likert scale.

7.3.4 Expected Governance Impact

Beyond the task-level constructs, respondents were asked where they expected the LACE Framework to make the greatest difference in data governance, regardless of task. The responses are summarised in Table 7.20 with the raw responses available in Appendix G.2. Most of the participants selected *standardization*, creating a unified format for data contracts. Three further areas were each selected four times out of six, namely *Automation*, *Codified Trust*, and *Purpose-binding of Data*. Allowing business users to draft contracts in natural language (*Accessibility*) was selected by half of the respondents, and no participants selected the *Other* option.

7.3.5 Qualitative Feedback on Suggested Improvements

The final open-ended question asked what respondents would change or add to make the framework more effective and practical. Four of the six participants provided suggestions; QP3 and QP5 left it unanswered. The responses are presented here first as a collection and then mapped onto the TAM3 constructs through deductive coding; their interpretation is addressed in the discussion (Chapter 8). The raw responses are available in Appendix G.2. Two points were raised by more than one respondent: how the framework models

Impact area	Count (of 6)
Standardization of heterogeneous data contracts	5
Automation of contract validation (CI/CD)	4
Codified trust (SLAs/SLOs/ownership)	4
Purpose-binding of data to authorized use cases	4
Accessibility for business users	3
Other	0

Table 7.20: Greatest expected data-governance impact; select-all-that-apply item derived from Table G.10. Answered by all six respondents.

data products and their constraints, and the consistency and reliability of its output. The remaining suggestions were raised by individual respondents.

Two respondents commented on how the framework models data products, their ports, and the constraints. QP2 suggested that the graph currently carries too much detail and provided a concrete example of a remodel, proposing that the freshness constraint be attached to the output port rather than represented as a separate entity. The participant additionally argued that the policy model is incomplete and should be extended with an expiration date and a business justification.

> QP2: *"The most important thing to avoid, I think, is information overload. I think the knowledge graph currently contains too much detail to be useful. Take, for example, the 'PromotionalAudienceConstraintFreshness' with relations to Currentness, operator left/rightOperand, and unit. In my view, it would be more useful to model Freshness as a constraint on the PromotionalAudience's output port.* [...] *On the policy side, you should go one step further: most policies also need an expiration date and a business justification."*

QP6 also made a related conceptual point about treating output ports as the consumed entity and referencing the Open Data Product Standard for input ports. QP6 also noted that the lineage would benefit from drill-down capability.

> QP6: *"Conceptually: one data product has one or more output ports. These output ports are the consumed entity. Use the Open Data Product Standard to define the input ports in YAML format.* [...] *lineage would benefit from a drill-down capability."*

A second point raised by two respondents was the consistency and reliability of the framework's output. QP1 was more concerned about consistency when similar data products are inserted, suggesting that the framework should check for similar existing data products before adding new ones to avoid near-duplicate definitions. QP1 also saw a benefit in the next step, in which the framework helps a non-technical user locate the right data product or, if none exists, define a new data product and contract to be added to the KG.

> QP1: *"In a large organization, I can imagine it becoming a problem that different data products coexist with slightly different definitions. To prevent this, the framework could check whether a similar data product already exists before adding a new one. A next step would be to help a non-technical person who is looking for an insight find the right data product, or, if it does not exist, to define a data product/contract so that it can then be added to the KG."*

QP6, on the other hand, raised a reliability concern regarding the trustworthiness of answers, questioning whether a query over the graph would return a fully reliable answer.

> QP6: *"If a data security lawyer asks 'In which data products are GDPR-relevant data fields?', is the answer really 100% reliable?"*

QP6 also noted that while the policy-code generation looks promising, they could not judge its usage or quality because their organization relies on a role-management system in which users apply for roles listed in contracts.

> QP6: *"Creating policy code looks nice, but I could not say anything about the usage nor the quality. In my organization, there is a role management system in place, so that you just need to apply for the roles mentioned in the contracts."*

QP2 also noted that the generated SPARQL query should rely on fuzzy matching, as in the demo environment, the participant sometimes received no results. Beyond the suggested change, QP2 described their organization's efforts to align the administration of data products and contracts with the application and data-domain landscape, and predicted that governance of KG data would become increasingly important.

> QP2: *"Prediction: Governance of knowledge graph data is going to become extremely important."*

QP4, by contrast, did not request a change but emphasized the framework's value for agentic workflows, in which the KG can provide focused context within a constrained token budget.

> QP4: *"I would emphasize the use case for agents; Agents require context, context limits constrain what they can be given, but the knowledge graph can optimize the provided data in a way better than other techniques, allowing us to build a "minimum viable context" containing the right data, at the right time, at the right token budget."*

In the end, the suggestions cover feedback on the model driving the framework, the consistency and reliability of the output, the query and policy functionalities, and the framework's value to agents.

Because no open-ended prompt was attached to each construct (see the threats to validity in Chapter 9), these responses cannot be claimed to have been organised by construct. They can, however, be mapped onto the constructs after the fact through deductive coding. We applied the same coding procedure used in the exploratory phase [75], but adopted a theory-driven rather than inductive approach, in which the TAM3 constructs are the a priori coding themes. Since the question was construct-agnostic, a single comment may bear on more than one construct. The coding is summarised in Table 7.21; because it reinterprets construct-agnostic responses, it should be read as an organising lens on the qualitative data rather than as a definitive construct-level measurement.

The coded feedback focuses on the QUAL and Self-Predicted Use constructs, the two constructs identified as weakest in Table 7.19. The QUAL cluster is the largest and spans both tasks. On the policy side, QP2 notes a missing expiration date and business justification, while QP6's output-port modeling maps to the lowest task score; on the retrieval side, QP2's empty SPARQL results and QP6's request for lineage drill-down indicate this task is not flawless either. The Self-Predicted Use codes are mostly organizational rather than technical, including QP1's near-duplicate detection and accessibility request, QP6's reliance on a separate role-management system, and QP2's note that their organization is still organizing its data-product administration. One comment, QP4's minimum viable context provisioning for agents, fits no TAM3 construct and is recorded as an emergent category.

Feedback	Participant	Construct	Note
Information overload / too much graph detail	QP2	QUAL + EOU	Granularity critique
Policy needs expiration date + business justification	QP2	QUAL (Policy)	Links to low QUAL in task 3
SPARQL should use fuzzy, multi-step matching	QP2	EOU + QUAL (Retrieval)	Negative on knowledge retrieval task
Organization still getting data-product administration in order; KG governance “will be extremely important”	QP2	Self-Predicted Use + USEF	Organizational readiness
Output ports as the consumed entity; use ODPS for input ports	QP6	QUAL (model fidelity)	Modelling mismatch
Lineage drill-down capability	QP6	QUAL (lineage/retrieval)	Feedback on better lineage
Policy-code usage and quality unassessable; organization uses a role-management system	QP6	IMPORT + Self-Predicted Use	Links to low IMPORT in task 3 and low adoption
GDPR query “100% reliable?” trust concern	QP6	QUAL + Self-Predicted Use	Quality about trustworthiness
Near-duplicate data products at scale; help non-technical users find or define products	QP1	USEF (scaling) + Self-Predicted Use	Accessibility/scale gap
Agent / minimum-viable-context provisioning	QP4	None	Emergent code outside TAM constructs

Table 7.21: Coding of the open-ended feedback onto the TAM constructs.

Chapter 8

Discussion

This research examined how data contracts can be enforced in decentralized data architectures, and, in particular, how LLMs can be used to translate heterogeneous contracts into a deterministic, machine-readable foundation on which enforcement can be automated. The research was guided by the DSR paradigm by Hevner et al. [49]. Eleven semi-structured interviews were conducted to understand the technical and organizational barriers to contract enforcement. Building on these findings, the study makes three contributions. First, it adds empirical, practitioner-grounded research to the sparse academic literature on data contracts. Second, it proposes a data contract ontology that provides a structured representation of contracts and their surrounding governance. Third, it instantiates the LACE Framework, an artifact that generates an intermediary semantic model from data contracts based on the ontology. These contributions were assessed through an ontology evaluation, an intrinsic extraction evaluation, and an extrinsic evaluation with six of the interviewed experts.

This chapter discusses and interprets the findings of this research and their contributions to existing theory. First, the results of the three evaluations are discussed and related back to the research questions in Section 8.1. The contribution to the academic literature is then discussed in Section 8.2, after which the practical implications for organizations operating decentralized data architectures are considered in Section 8.3. In Section 8.4, we discuss different application scenarios in which the artifacts can be implemented and used.

8.1 Discussion of Findings

Taken together, the three evaluations answer the main research question, *how can LLMs help to enforce data contracts*, not by showing that an LLM can enforce a contract directly, but by showing that an LLM can reliably generate a deterministic intermediary model on which enforcement depends. The intrinsic evaluation establishes under which conditions the construction is accurate and reproducible (**SRQ5**), the ontology evaluation establishes that the resulting model is structurally sound and answerable (**SRQ4**); and the extrinsic evaluation establishes where practitioners perceive the model to deliver value. The findings below are interpreted individually and then related to one another.

8.1.1 Ontology Evaluation

The ontology evaluation demonstrates that the intermediary model is not only structurally valid but also capable of extracting answers that meet realistic information needs, and that it addresses the components required for automating governance (**SQR4**). Each of the three stakeholder scenarios is resolved using SPARQL queries, with no external parties involved. The single OOPS! The pitfall was informative and resolved by removing the inverse relationship to preserve consistency, but it introduced a new minor pitfall that was ignored. The remaining flagged pitfalls came from imported ontologies and fall outside of the scope of this work.

8.1.2 Intrinsic Evaluation

8.1.2.1 Structural Conformance

A single mechanism underlies every intrinsic result: the strategies that fail do so through structural conformance rather than syntax. Each evaluation, F1-score, Jaccard Index, SHACL validity, and entropy variance, shows that each configuration can be placed in one of two clusters: the example-based strategies (one-shot and OS-CoT) and the strategies without an example (zero-shot and CoT). That all four agree on the partition points to a single capability, producing structurally sound triples, as the separator. The triple counts confirm this is a quality rather than a quantity problem, as all configurations generate a similar amount of triples ($200-255$), while the exact matches differ greatly (YAML direct: zero-shot $18/255 \approx 7\%$ and one-shot $139/229 \approx 60\%$). The error analysis shows where it goes wrong: the dominant failure mode is SHACL violations rather than syntax, and the errors are mostly concentrated in the ODRL policy model and datatype discipline regions of the ontology. This shows that the model attaches the right vocabulary in the wrong shape.

8.1.2.2 Prompting Strategies

An interesting finding about the different prompt strategies is that "examples beat reasoning" is an understatement, because it is mostly an interaction effect. In direct mode, where entities and relationships are extracted in one single pass, a single example suffices, and one-shot is the strongest configuration. The split mode loses this single-pass coherence of having generated those entities itself. The relation stage must therefore form triples over an entity set that it did not produce. By explicitly specifying how the LLM must reason, it compensates by re-deriving the link in the relationship-extraction call. The results are clear: OS-CoT barely degrades (YAML: $0.773 \rightarrow 0.730$) while one-shot decreases significantly (YAML: $0.781 \rightarrow 0.554$) since the example shows the output shape but not the reasoning. A second factor compounds this: each of the two stages is an independent generation call, subject to its own syntax and SHACL failures, increasing the likelihood of generating malformed triples, as shown in the results for token usage and execution time. What stays true is that some scaffolding is required, as neither zero-shot nor standard CoT reaches usable accuracy under any configuration. The example does the one thing that reasoning alone cannot do, while reasoning helps the LLM understand the previous context. These results align with Martin et al. [70], who found that example-based prompting was most effective for triple generation, while zero-shot yielded low performance. The comparison between the results will be further discussed in Section 8.1.2.5.

8.1.2.3 Determinism

The gap between the exact and fuzzy Jaccard Indices is low for the stronger strategies (e.g., OS-CoT YAML direct: exact - 0.795, fuzzy - 0.969), but large for the weak ones (e.g., zero-shot text direct: exact - 0.351, fuzzy - 0.877). It is important to note that a high surface stability (high Jaccard Index) does not mean that it is correct. The high fuzzy Jaccard Index for the non-example prompts shows that the configurations reproduce a similar but wrong set of triples across runs. The same caution applies to the similar precision and recall scores. The weak strategies, where $P \approx R \approx 0.14-0.25$, are a symptom of structurally wrong output rather than balanced errors. OS-CoT is therefore the most deterministic prompt strategy on both indices and the only one whose reproducibility comes with high accuracy. Both of these properties are required to provide a deterministic foundation for data contract enforcement (**SRQ3**).

8.1.2.4 Structural Validity and Entropy

Since a large number of failures are structural rather than semantic, SHACL validation is the core component of the verifier module, intercepting the errors the LLM is most prone to. The entropy analysis reinforces this, but only when the mean number is compared with the mean variance, as the mean entropy ratio is high across configurations ($0.788 - 0.836$). This suggests that the graph's core is mostly governance-related, but since most of the violations surround the SHACL shapes, a high mean can equally reflect a malformed structure, so the mean cannot distinguish a valid graph from an invalid one. The discriminating value is the variance, as the example-based strategies produce compact distributions, whereas the zero-shot and CoT prompts produce many outliers, including nine extractions in which no governance chain is recovered. Notably, this is the only dimension in which splitting actually helps: it degrades accuracy, determinism, and validity across most configurations, although it produces fewer zero-chain failures and yields a more stable entropy distribution.

8.1.2.5 Comparison with Related Extraction Work

The framework's results can be compared with those of Martin et al. [70], who evaluate ten prompting strategies across six open models on a log-to-RDF extraction task. Three of their findings match our research. First, contextual examples significantly improve extraction, with their few-shot prompt strategy achieving the best results across nearly all models (Llama few-shot reaching an F1-score of 0.9935), while zero-shot prompts reached at most 0.30. This matches the gap we also see in our results between the non-example and example-based prompt strategies. Second, reasoning is not a primary improvement on the prompt strategy, as there are more advanced reasoning strategies that underperform example-based strategies and few-shot with chain-of-thought barely exceeds few-shot alone, again consistent with our findings that reasoning instructions matter only at the margin (under direct extraction). Third, their separation of syntactic from relaxed-semantic scoring reveals the same failure mode we call structural conformance. In Martin et al.'s research, syntactic-semantic failures occur with numerical, resource, and identifier predicates that are semantically correct but formatted incorrectly, similar to our finding of attaching the right vocabulary in the wrong shape.

Besides the matching results, there are also differences between the two studies. Their absolute scores far exceed ours (best F1 ≈ 0.99 against ours ≈ 0.78), but do note that the tasks are not comparable, as their pipeline maps a single log line to a flat, closed predicate set with a one-to-one gold standard, whereas ours populate a multi-module ontology with ODRL policy constraints from different structured and unstructured files. The second divergence is the type of prompts used: our research includes no few-shot prompting, as we did not go beyond one example, whereas their research shows that multiple examples yield better output. Their few-shot with CoT prompt strategy barely improved results over the standard few-shot prompt, whereas our OS-CoT strategy is the most robust under split extraction, suggesting that reasoning is preferred when the global context is fragmented across passes.

A second point of comparison is the work of Huang et al. [53], who constructed KGs from unstructured marine accident reports with an LLM pipeline and, like this research, adopt a combination of CoT and one-shot prompting. Their results reinforce our intrinsic results, as their absolute performance is close to ours: their LLM-based method achieves an average F1 score of 0.767, compared to our best configuration of 0.781. Both theirs and ours result from operating the pipeline on heterogeneous, real-world documents rather than on a closed, single-line extraction task like the one in Martin et al. This similarity suggests that an F1 in the high 0.70s is the current capability of LLM extraction over unstructured sources without fine-tuning.

A third point of comparison concerns the extraction failures. Cotti et al.'s OntoLogX extracts ontology-grounded KGs from cybersecurity logs and, like the verifier module in this work, closes its pipeline with a generation, validation, and correction loop. Each generated

graph by the LLM is checked for syntactic validity, then for SHACL conformance, and if it is violated, then it is fed back into the model for iterative repair [21]. Their evaluation, in contrast to ours, spans eight LLMs, including a Mistral Large model. The final results in their research mirror ours, once the correction stage is active, SHACL violation ratios drop in contrast to their baseline runs which confirms that shape validation is the mechanism that makes LLM output reliable and usable. While the corpus and domain are widely different, Cotti et al. show that having a validation stage is a necessity in an ontological LLM-based extraction pipeline.

Diving deeper in their results, it shows more similarity to our research results. First, structured output alone proved insufficient for generating valid graphs, as it frequently produced malformed graphs, and became only usable when provided with examples [21]. This mirrors our results that examples, not reasoning or structure on their own, separate the usable configurations from the unusable ones. They also observe a divergence between semantic and ontological quality where example-free configurations reach high semantic scores while scoring low on F1, because the model extracts the correct information, but introduces noise that lowers precision and breaks structural conformance often [21]. This is the same caution raised in our research around the gap between the fuzzy and exact Jaccard Indices, where high similarity can mask structurally incorrect output.

8.1.3 Extrinsic Evaluation

The extrinsic evaluation shows an overall positive reception of the LACE Framework, but the internal ordering is more informative than the overall scores. The tasks ranked consistently across the constructs, with knowledge retrieval and querying ranked highest and executable policy generation lowest. The expected governance impact responses corroborate this, with standardization ranking highest, suggesting that respondents see standardization as a prerequisite for automated enforcement rather than the final result. The contrary perspective by respondent QP6 is worth isolating. QP6's objections (organizational inertia, overhead of breaking down tacit knowledge, and immature vendor tooling) target the organizational and immature landscape required to operate an ontology layer, not the conceptual soundness.

The deductive coding of the open-ended feedback (Table 7.21) shows two things the Likert scores miss. First, it provides validity through an independent method, as the codes concentrate on the same constructs the Likert scores identify as weakest. Second, it separates technical barriers (graph complexity, policy incompleteness, retrieval fuzziness) from organizational ones (role management, near-duplicate governance, administrative readiness). This shows that the adoption gap is mostly attributable to organizational rather than technical concerns, which is why USEF and ATT remain positive while Self-Predicted Use alone falls on the disagreeing side. The technical barriers that do surface are not objections to the concept but signs that the product is not yet viable in the respondents' own context and that some flaws need fixing before it is production-ready. One comment resists the frame entirely: QP4's view of the framework as a context provider for autonomous agents, supplying a "minimum viable context" at the right token budget, fits no TAM3 construct and points to a usefulness dimension, LLM-agent context provisioning, that the survey never measured.

8.1.4 Relating the Intrinsic and Extrinsic Findings

While intrinsic evaluation measures extraction accuracy against a gold standard, extrinsic evaluation assesses practicality based on six experts' perceptions of the demonstration environment. Between the two, there are some similarities, one of which is the policy layer, which is the weakest link in both evaluations. Intrinsically, it is the largest source of SHACL violations, while extrinsically, it is rated the lowest on EOU and QUAL. The same component is structurally fragile and is also poorly received. The exact opposite is true for knowledge retrieval: it is rated highest for scores, EOU, and QUAL, and the ontology evaluation shows

that it works. Two notes have to be made. First, the practitioners assessed a curated demonstration environment built with the gold standards, so their perceptions reflect best-case extraction scenarios rather than the average extraction. Second, the Self-Predicted Use score is not fully explained by extraction quality, as USEF and ATT remain positive while the adoption intention drops significantly.

One limitation of the artifact itself is worth noting alongside these findings. The framework was designed to keep the intermediary model aligned with evolving contracts (**FR4**), but the current instantiation only works through merging and cannot remove triples. Additions and extensions are therefore reflected, whereas entities and relations that a contract no longer contains persist inside the model. **FR4** is therefore met for the growth of the model but not for its full converge with a contract's current state.

8.2 Academic Implications

This research produces a composite artifact, and each of its three parts contributes to the academic literature on data contracts, a topic that lacks mature standards and empirical studies.

The first artifact is the exploratory knowledge produced in the interview phase. It contributes a practitioner-grounded definition of a data contract and a relational theory of the conditions, strategies, and consequences surrounding its adoption, addressing the organizational and technical barriers of **SRQ1** and the current state of industry implementation in **SRQ2**. While academic literature largely draws on gray literature, this research complements it with a systematic, theory-grounded perspective. This positioning is supported by the academic literature itself. Wasser et al.'s systematic review of data contracts concludes that scholarly work on the topic remains limited and that the concept is still defined through practitioner (gray) literature [103]. This research complements their gray literature review, since the data contract research challenges they derive in answer to their RQ4 (*What Are the Research Challenges in Data Contracts?*) correspond to the three artifacts produced here. First, they mention there is no standardization due to a lack of conceptual models like an ontology for data contract-based systems, which the Data Contract Ontology addresses (artifact 2). Second, they call for an NLP or LLM-based method to translate natural language data contracts into contract specification, which the LACE Framework realizes (artifact 3). Finally, they explicitly mention doing a qualitative study with practitioners to surface their view on data contracts, which our exploratory phase provides (artifact 1). Their remaining challenges, automated contract enforcement and contract lifecycle management, map onto the downstream applications which can be built on top of the Semantic Module [103].

The second artifact is the Data Contract Ontology. It integrates and extends several existing vocabularies into one representation of contracts and products. This ontology provides a reusable semantic foundation for decentralized data architectures.

The third and final artifact is the LACE Framework, which is as much a methodological contribution as it is a technical one. The LACE Framework, with its evaluations, demonstrates the difficulty of applying LLMs to a governance-extraction task in which structural conformance is important. In this research, it is shown that this difficulty is best addressed by combining reasoning with examples and shape-based validation rather than by reasoning prompts alone. This finding is transferable beyond data contracts to any setting in which LLM output must populate a constrained ontology.

Hechler et al. envisioned an *AI-infused automated AI governance insight component* built from KG inclusion, ontology mapping, and text analysis of unstructured regulatory documents, but explicitly note that this scope is "not a reality yet and subject to R&D" [48]. The LACE Framework instantiates and evaluates the extraction and mapping stage of exactly that component, providing evidence of both its feasibility, a high F1-score and its perceived usefulness to practitioners, and its main obstacles, structural conformance and the ongoing

refinement of the ontology model with practitioners.

Quantitatively, these extraction results are competitive with comparable LLM-based knowledge-graph construction studies over unstructured sources.

8.3 Practical Implications

Only two of the three artifacts have practical implications: the Data Contract Ontology, as the layer that organizations would adopt to standardize and query their data products and contracts, and the LACE Framework, as the pipeline that populates the Semantic Module using the Data Contract Ontology.

The first artifact, the ontology, is best understood as a standardisation and discovery layer before being understood as an enforcement layer. Both the practitioner evaluation and governance impact responses show that the greatest value lies in the ontology layer: a single machine-readable representation of heterogeneous contracts and the ability to make that representation queryable. Organizations that would adopt it should therefore expect the earliest returns in data discovery, lineage, and contract registration, rather than in fully automated policy enforcement. Expressing the model in RDF enables reasoning, as inference rules can derive facts that were never explicitly stated. For instance, when a column is annotated as PII, a reasoner can propagate that classification along the lineage to every product that uses it. This means that downstream and newly added products automatically inherit the privacy obligations. The RDF reasoner could also derive that a policy is internally inconsistent when the same party is both permitted and prohibited the same action on the same target. Such a contradiction, which appears only when combined, reveals a governance conflict not stated in the contract.

The second artifact, the LACE Framework, yields a technical artifact and two concrete deployment recommendations. As a technical artifact, it is a reusable extraction-and-validation pipeline that organizations can run to populate the Semantic Module from their existing structured and unstructured data contracts. Beyond the pipeline itself, it produces two deployment recommendations. The first is that examples are more important than reasoning prompts, especially for direct and split extractions; a combination would work better. Second, the extraction often fails due to structural issues, so a validation step is crucial for governance assurance. This research shows that the SHACL validation stage helps the LLM, as a probabilistic text generator, produce a deterministic foundation.

These implications can be extended with the closest comparable system in the literature: Wider et al.'s Governance AI [105]. In their research, they use an LLM to remove the manual translation of policy documents into rule-based contracts. Our research differs in where the LLM sits: they place it in the decision loop, evaluating access requests directly against plain-text policies and emitting warnings, whereas we place it outside the decision loop, using it only to translate contracts into RDF triples and populate a deterministic intermediary model. Both approaches still keep a human in the loop, converging on the same governance principle that accountability cannot be fully delegated to the model.

8.4 Application Scenarios: Organizational Use of the LACE Framework & the Semantic Module

While the intrinsic and extrinsic evaluation demonstrate the technical viability and perceived usefulness of the LACE Framework, its ultimate value lies in solving concrete organizational bottlenecks. The following applications illustrate how the proposed artifacts transition from theoretical constructs to operational assets in a real organizational architecture. It is important to note that these applications are illustrative and represent envisioned implementations.

8.4.1 Extraction Standardization

In large organizations, different domains exhibit varying levels of technical knowledge and can impact the way how data contracts are written. Software engineers might be comfortable writing contracts in strict, version-controlled YAML, while non-technical domains may rely on unstructured formats such as plain text or Excel. The LACE Framework can be deployed as an ingestion layer within a self-serve data platform to standardize this extraction process. Rather than forcing non-technical domains to conform to a rigid technical syntax, the organizations allow domains to submit contracts in their preferred formats, optimizing employee experiences on the platform. As the platform experience improves, users are motivated to remain within the platform's governed ecosystem. This eliminates the need to use ungoverned tools to achieve their goals, thereby minimizing the likelihood of shadow behavior. By acting as a seamless translation layer, the framework parses the documents into a single, unified representation. This ensures that extensions built on top of the Semantic Module receive a structured and predictable format without losing meaning or structural integrity.

8.4.2 Digital Twin

As decentralized data architectures scale, cross-domain dependencies become increasingly complex, creating friction between centralized control and local speed. Once the Semantic Module is populated with the LACE Framework (and its corresponding data product information), the resulting RDF KG effectively serves as a digital twin of the data mesh. Consequently, the entire organization's data topology becomes deterministically queryable. Such a structural map of the data mesh makes invisible dependencies visible, allowing domains to see how a major version change to their data products affects downstream consumers. More importantly, automated continuous integration can query the graph to instantly identify all downstream consumers and SLAs that might be violated by a change before any reaches production, thereby enabling proactive analysis and mitigating cross-domain friction.

8.4.3 Semantic Reasoning

Legal and organizational mandates require strict access controls, but translating human-readable policy agreements into technical configurations often creates an operational bottleneck. By expressing data contracts as an RDF graph (Semantic Module) and aligning them with the ODRL-based policy modules (Data Contract Ontology), the organization unlocks RDF reasoning capabilities. The graph natively understands the relationships between assets, duties, constraints, permissions, and prohibitions. Instead of viewing the Semantic Module as a passive catalog, organizations can use it as an active computational governance engine. For example, if a data contract explicitly prohibits the external distribution of an asset, the reasoning engine can automatically infer that downstream consumer applications lacking internal clearance must be denied access. This transitions the Semantic Module from a descriptive tool to the first-line enforcer of federated governance.

Chapter 9

Threats to Validity

Several limitations affect the validity of the research. They are grouped below into threats to internal validity, which concern the research process and decision-making; construct validity, which concerns whether the measures capture what they intend to; and external validity, which concerns how far the results generalize beyond the study's setting.

9.1 Internal Validity

Several decisions during the research process affected the internal validity. The qualitative analysis evolved during execution: the initial intention was to combine Reflexive Thematic Analysis with the Leuven Qualitative Analysis Guide (QUAGOL), but this was abandoned because the per-interview documentation proved too time-consuming. The analytical lens later shifted toward a more grounded, relational analysis structure when it became apparent that descriptive themes alone could not capture the causal and contextual relations between codes.

The interview guide was also refined during data collection: questions initially branched based on the participant's architectural experience, but were combined into a single set once they proved redundant, an adaptation consistent with the view that adapting an interview is a legitimate part of improving it [3].

The final instrumentation gap is that the open-ended feedback questions were not specifically attached to each TAM3 construct. This limits the explanatory detail available behind the quantitative score and means that participants were not asked to reflect on each task immediately after completing it, when their impressions were still fresh.

9.2 Construct Validity

The extrinsic evaluation relied on a TAM3-based survey, and two construct-level limitations apply. First, the reliability analysis returned one item set below the conventional threshold (task 2 EOU with $\alpha = 0.659$) and one negative coefficient (task 2 QUAL with $\alpha = -0.706$), although the latter is a known small sample size artifact arising when responses are similar with low variance rather than evidence of an unreliable question set [99]. Both these values indicate that the corresponding task 2 scores should be interpreted with caution. This limitation is quite impactful, as task 2 is the best-rated task; although the score magnitude might not be reliable, the ordering is. In both EOU, QUAL, and IMPORT, it scores most favourable, meaning that all constructs mention that this is the most valuable task.

More importantly, the Self-Predicted Use is a proxy for actual adoption rather than a measure of it. This is because participants assessed the framework using guided tasks in a sample environment rather than in their own environment. This means that the judgements from the participants are somewhat speculative, which is also mentioned by QP6 in the open-ended feedback.

A final validity threat concerns the intrinsic benchmark itself, as the gold standard was produced by a single annotator, so inter-annotator agreement (e.g., Cohen's kappa) could

not be computed, and the reference triples may reflect a single interpretation. We tried to minimize this risk by constraining the annotation to the ODCS structure rather than free interpretation.

9.3 External Validity

A significant validity concern is the generalization of the LACE Framework. The framework was built and assessed in a constructed demonstration environment rather than in a live, production data mesh. This means that under real operational scale with heterogeneous data contracts, the evaluations might differ from what we report here.

Relatedly, the evaluation environment was scoped to a one-to-one mapping between data products and data contracts. In practice, a single data product may be governed by several contracts (for example, distinct contracts per output port or per consumer agreement), and a contract may span products. This simplification keeps the gold-standard construction and the intrinsic benchmark tractable, but it leaves the framework's behaviour under many-to-one and one-to-many contract relationships untested.

The extrinsic evaluation draws on six of the eleven industry experts who were interviewed during the exploratory phase. This sample size is too small to support statistical inference and only indicates direction and flags divergent items. This final extrinsic evaluation was the only one conducted with experts, which limits the maturity of the instantiated model being judged.

Finally, while a data contract ontology was developed during this research, it was assembled largely from reused external vocabularies rather than engineered from first principles due to time constraints.

Chapter 10

Conclusions

The manual translation of data contracts into executable enforcement mechanisms is a bottleneck for scalability and consistency in decentralized data architectures. Technical engineers need to interpret heterogeneous, human-authored contracts into code and tool configurations that do not scale across a growing data landscape. Following the DSR paradigm by Hevner et al. [49], this thesis addressed that bottleneck by investigating how LLMs can construct a deterministic, machine-readable foundation on which enforcement can be automated. The resulting artifact is a composite instantiation of three parts: practitioner-grounded exploratory knowledge, a Data Contract Ontology, and the LACE Framework that populates an intermediary model.

The remainder of this chapter is organised as follows. Section 10.1 answers each research question, addressing the five sub-research questions before answering the main research question. Section 10.2 then sets out possible directions for future work, spanning ontology, the underlying graph technology, database-to-graph mapping, and more.

10.1 Answering the Research Questions

SRQ1 — What are the primary technical and organizational barriers for enforcing governance within decentralized data product architectures? Through the exploratory interviews, two main barriers were found (described in Section 4.4).

The technical barrier shows the absence of a standardised contract specification that captures technical, legal, and business semantics in a single, machine-readable format. Because organizations prioritise semantics over syntax, these governance and legal constraints reside inside the contract, and technical teams are left to translate them into executable enforcement code or into enforcement tools manually. This manual translation is the immediate scalability and consistency bottleneck, and it can introduce translation loss where business intent or legal compliance is lost or misinterpreted as the number of data products grows.

The second barrier, organizational and regulatory, is even more critical and explains why this translation and enforcement can not be handed over to an LLM. As the legal and geopolitical context shows, some regulated environments require a human-in-the-loop. This requirement limits the use of AI, whose non-deterministic behavior experts regard as incompatible with governance decisions. Organizationally, the decentralized data architecture industry remains immature: fragmented tooling and unstable standards keep enforcement a substantial engineering effort, and organizational inertia further slows adoption.

Together, these barriers explain why governance is hard to automate and enforce consistently throughout an organization.

SRQ2 — How are data contracts currently implemented and enforced in the industry? This question was addressed through the exploratory interviews as well, and it was found that although practitioners regard data contracts as an essential part of a decentralized data architecture, their implementation in practice is inconsistent, and their enforcement is frequently absent.

Where contracts do exist, they are often not in a standardized format, as organizations prioritise semantic meaning over the document's syntax. This deliberate choice is made so that business users can also participate, and because a single contract should capture not only technical SLAs and quality guarantees but also ownership, consumer obligations, purpose-binding rules, and privacy classifications.

Data contract enforcement, when it is attempted at all, relies on a mix of technologies and is often reduced to the data catalog with manual access control, or a variety of different (immature) tools. As a result, engineers remain responsible for translating the legal and business rules in each contract into enforcement code, a task that does not scale and yields inconsistent enforcement across a growing data landscape.

SRQ3 — What architectural principles and mechanisms are required to ensure the trustworthiness and reliability of LLM-extracted data contract information? The requirements for trustworthiness were derived from the exploratory interviews, and the mechanisms to satisfy them were validated during the intrinsic evaluation. Industry experts noted that because LLMs are non-deterministic and often operate in regulated production environments, their output should not be trusted blindly. Another principle is keeping accountability; this means that a human-in-the-loop must be possible, and that every extracted statement must be traceable to its original source for verification.

To address the non-deterministic nature of LLMs, SHACL validation mechanisms were added to the pipeline, providing a safeguard that ensures structural validity. Human-in-the-loop and traceability are addressed through the provenance lineage model. This model preserves the origin and the time at which each statement is generated, thereby enabling human verification.

The determinism analysis shows that example-based prompting strategies achieve high reproducibility across runs (Jaccard Index ≈ 0.79), enabling consistent output.

Trustworthiness and reliability, therefore, rest on a division in which example-based prompts enable semi-reproducible extractions, SHACL validation ensures structural validity, and provenance modeling supports human-in-the-loop and traceability requirements.

SRQ4 — What components and interactions are required to automate the governance surrounding data contracts & products? This research question was addressed by designing the LACE Framework in Chapter 5 and instantiating the design principles in Section 4.5. Automating governance does not require directly enforcing data contracts; it requires transforming these heterogeneous contracts into a machine-readable, standardized foundation. This foundation is one of the three required components and serves as the basis for downstream automation.

Before the foundation can be populated, a format-agnostic parser needs to be available. Because data contracts lack a mature standard and exist in various file types, the parser needs to analyze these formats. In our research, we have done this through an LLM-assisted parser.

The second and central component is the foundation itself: the Semantic Module, into which the parser writes. Its role is standardization. By expressing the contract as RDF against an ontology, validated through SHACL shapes, it replaces the documents with a deterministic model of the data products and their governance rules. The ontology and its shapes are the schema and the integrity layer, which makes this model machine-readable, but this is not the automation itself.

The third component is an easily accessible endpoint on the Semantic Module, such as an API, that exposes the standardized Semantic Module to the outside world. This is what makes the foundation usable as an access point, allowing extensions to be built on top of the standardized, deterministic foundation. Through such an endpoint, the topology and constraints can be queried and translated into automation of governance, such as generating policy code or populating a data catalog.

SRQ5 — What is the impact of different prompt engineering techniques on LLM-extracted data contract information? Through intrinsic evaluation, this question was answered in Section 7.2.1. The four different prompt strategies (zero-shot, one-shot, CoT, and OS-CoT) are evaluated based on accuracy, structural validity, determinism, efficiency, and the governance chain.

The main finding is that providing an example is what separates usable from unusable output: example-based prompts achieve an F1 Score above 0.7 in direct mode, whereas example-free prompts never become usable. Adding explicit reasoning steps mitigates the loss of context during split extraction, where one-shot drops to 0.554 while OS-CoT holds at 0.730.

The same ordering holds for the other dimensions, with OS-CoT achieving up to 100% SHACL validity and incurring the fewest violations overall. One-shot comes close behind, but the zero-shot and CoT prompt strategies fail the majority of validation rounds and worsen under splitting. Regarding determinism, the example-based strategies are highly reproducible (exact-match Jaccard Index around 0.79 in direct extraction mode), whereas prompts without examples fall in the range of $0.25 - 0.41$. Regarding efficiency, OS-CoT in direct mode is the cheapest in terms of overall token consumption and the fastest configuration, largely because weaker strategies trigger more repair cycles than OS-CoT. The entropy ratio adds little on its own, as the mean is high and nearly flat across all configurations; the variance discriminates, with example-based prompts producing compact distributions and example-free prompts producing many outliers.

Prompts have a definite impact on the performance of the extracted information. The one-shot strategy maximises peak accuracy, while OS-CoT is the most robust and efficient strategy and the only one that combines high accuracy, high validity, and high reproducibility across both extraction modes. Prompts without examples are never usable for this task across all dimensions.

Main Research Question — How can LLMs help to enforce data contracts? LLMs help enforce data contracts not by enforcing them directly, but by acting as a translation layer: they convert heterogeneous, human-authored contracts into a deterministic, machine-readable intermediary model on which downstream enforcement or an agent traversing the graph can build. The LLM's contribution, therefore, sits upstream of the enforcement decision: it resolves the manual translation bottleneck identified in SRQ1 and SRQ2 by converting unstructured and structured contracts into a standardized representation.

The three evaluations together substantiate this answer. The ontology evaluation shows that the intermediary model is structurally sound and expressive enough to answer realistic needs. The intrinsic evaluation establishes the conditions under which an LLM can reliably populate the model, and the extrinsic evaluation shows that practitioners recognize the value, rating standardization and knowledge retrieval as highest, while identifying the main adoption barriers as organizational rather than technical concerns.

This holds only under three conditions: the model is constrained, its output is validated against an explicit schema, and its generation is reproducible. Within these bounds, the LLM provides a trustworthy foundation for constructing the governance model, but not for making enforcement decisions on its own. The evaluations also reveal where the approach is weakest: the policy region of the ontology.

In short, the LLM's role is to make data contracts machine-readable and queryable at scale, transforming a manual, error-prone translation step into a validated and repeatable process. This standardized, deterministic foundation enables downstream automation, from generating executable policy code and populating data catalogs to providing focused context for autonomous governance agents, and is the central contribution of this thesis.

10.2 Future Research

This research opens several directions for future work. They follow from the components we left specified rather than built, and from the weaknesses our evaluations surfaced.

- **Strengthening the Data Contract Ontology.** The ontology was the weakest artifact in both the intrinsic and extrinsic evaluation, concentrated in the ODRL policy region that produced the most SHACL violations during extraction and that the experts rated lowest on Ease of Use and Perceived Output Quality. Future work could iterate on the ontology together with domain experts to increase its practical usability and strengthen the policy component used for governance automation. It could further be extended by integrating a mapping language such as RML [55] or D2RQ to map data from the operational database directly into the Semantic Module. Combined with Ontology-Based Data Access, this would allow a virtual knowledge graph to be queried against live data without materialising it [108], narrowing the gap between the contract and the data it governs.
- **Prompt engineering.** This research tested only single-example prompts. Adding more examples could further increase the reliability and performance of the extraction, in particular the validity percentage for the policy parts of the ontology.
- **Underlying graph technology.** RDF was chosen as the graph model for its reasoning capabilities, but future work could compare it against a semantic property graph [82] or a labelled property graph backed by a constraint schema such as PG-Schema [10, 83]. Such a comparison would show which representation best balances expressiveness, validity, scalability, and tooling for data contract enforcement.
- **End-to-end knowledge graph construction.** This research assumed that a structural graph was already in place and that only the information from the data contracts had to be extracted. A complete pipeline would build the data-product graph and all its constraints in a single process, automating the creation of the Semantic Module and testing the LACE Framework's generalizability across a larger portion of the data mesh.
- **Fine-tuning the LLM.** Fine-tuning an LLM with Low-Rank Adaptation (LoRA) could give it the domain knowledge needed to predict RDF triples more accurately [44]. Since the error analysis showed that structural conformance, rather than content retrieval, is the dominant failure mode, this kind of fine-tuning could improve the part where extraction is most challenging.
- **Agent-based traversal of the Semantic Module.** The endpoint specified for the Semantic Module opens a direction that emerged from the expert feedback: the intermediary model could serve as a context source for autonomous agents that traverse the knowledge graph to answer governance questions. Because agents are limited by their context window, a standardized graph provides a way to supply the right information within a constrained token budget, making agent-based traversal a natural extension of the foundation this thesis provides.
- **Full graph synchronization.** The current update mechanism is insert-only, meaning that removed information in the contract still persist in the Semantic Module. Future work could add a differencing step that compares the extraction and the KG, deleting the triples a contract no longer has. This allows the graph to synchronize the data contract's current state.

Use of AI Tools

During the preparation of this thesis, AI tools were used to support the writing process. These tools were used for language editing, rewriting passages for clarity, checking the text for consistency and errors, assisting in the creation of diagrams and tables, and helping in the development of code. All research design, implementation, analysis, results, and conclusions are my own. Every AI output, including the generated text, code, diagrams, and tables, was reviewed and verified for accuracy before being included, and I take full responsibility for the content of this thesis.

Bibliography

[1] ATLAS.ti. 20

[2] Muhamad Abdurahman, Fariz Darari, Hans Lesmana, Muhtar Hartopo, Immanuel Rhesa, and Berty Chrismartin Lumban Tobing. Lex2KG: Automatic Conversion of Legal Documents to Knowledge Graph. In *2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 1–5, October 2021. ISSN: 2330-4588. 12

[3] William C. Adams. Conducting Semi-Structured Interviews. In *Handbook of Practical Program Evaluation*, pages 492–505. John Wiley & Sons, Ltd, 2015. Section: 19 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119171386.ch19. 18, 19, 126

[4] Omolola A. Adeoye-Olatunde and Nicole L. Olenik. Research and scholarly methods: Semi-structured interviews. *JACCP: JOURNAL OF THE AMERICAN COLLEGE OF CLINICAL PHARMACY*, 4(10):1358–1367, 2021. _eprint: https://accpjournals.onlinelibrary.wiley.com/doi/pdf/10.1002/jac5.1441. 18, 19

[5] Nachman Agmon and Niv Ahituv. Assessing Data Reliability in an Information System. *Journal of Management Information Systems*, 4(2):34–44, September 1987. _eprint: https://doi.org/10.1080/07421222.1987.11517792. 1

[6] Khlood Ahmad, Mohamed Abdelrazek, Chetan Arora, Muneera Bano, and John Grundy. Requirements engineering for artificial intelligence systems: A systematic mapping study. *Information and Software Technology*, 158:107176, June 2023. 22

[7] Sarat Ahmad, Zeinab Nezami, Maryam Hafeez, and Syed Ali Raza Zaidi. Benchmarking Vector, Graph and Hybrid Retrieval Augmented Generation (RAG) Pipelines for Open Radio Access Networks (ORAN), July 2025. arXiv:2507.03608 [cs] version: 1. 59

[8] Ibrahim Alhassan, David Sammon, Mary Daly, Arif Wibisono, Laleh Kasraian, Tadhg Nagle, Ciara Heavin, Denis Dennehy, Efpraxia Zamani, and Alaa Qaffas. THE USE OF OPEN, AXIAL AND SELECTIVE CODING TECHNIQUES: A LITERATURE ANALYSIS OF IS RESEARCH. *UK Academy for Information Systems Conference Proceedings 2023*, June 2023. viii, 21, 30

[9] Mohammad Mulayh Alshammari and Yaser Hasan Al-Mamary. User acceptance of AI-powered training: extending the technology acceptance model (TAM). *Future Business Journal*, 11(1):239, October 2025. 26

[10] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. PG-Schema: Schemas for Property Graphs. *Proc. ACM Manag. Data*, 1(2):198:1–198:25, June 2023. 131

[11] Mihir Athale and Vishal Vaddina. Knowledge Graph Based Repository-Level Code Generation. In *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, pages 169–176, May 2025. 12

[12] Sunil Bhalla, Nurhidayah Bahar, and Kanagi Kanapathy. Pre-testing Semi-structured Interview Questions Using Expert Review and Cognitive Interview Methods. *International Journal of Business and Management*, 7:11–19, October 2023. 18

[13] Bitol. Open Data Contract (ODCS), 2025. viii, 2, 45, 50, 53, 63

[14] Ivo Blohm, Felix Wortmann, Christine Legner, and Felix Köbler. Data products, data mesh, and data fabric. *Business & Information Systems Engineering*, 66(5):643–652, October 2024. 1

[15] Jan Bode, Niklas Kühl, Dominik Kreuzberger, and Carsten Holtmann. Toward Avoiding the Data Mess: Industry Insights From Data Mesh Implementations. *IEEE Access*, 12:95402–95416, 2024. 6

[16] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, January 2006. _eprint: https://doi.org/10.1191/1478088706qp063oa. 17

[17] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, 2020. Version Number: 4. 10, 80

[18] David Byrne. A worked example of Braun and Clarke's approach to reflexive thematic analysis. *Quality & Quantity*, 56(3):1391–1412, June 2022. 19, 20

[19] Steve Campbell, Melanie Greenwood, Sarah Prior, Toniele Shearer, Kerrie Walkem, Sarah Young, Danielle Bywaters, and Kim Walker. Purposive sampling: complex or simple? Research case examples. *Journal of Research in Nursing*, 25(8):652–661, December 2020. 18

[20] Seungmin Choi and Yuchul Jung. Knowledge Graph Construction: Extraction, Learning, and Evaluation. *Applied Sciences*, 15(7), March 2025. 24

[21] Luca Cotti, Idilio Drago, Anisa Rula, Devis Bianchini, and Federico Cerutti. OntoLogX: Ontology-Guided Knowledge Graph Extraction From Cybersecurity Logs With Large Language Models. *Advanced Intelligent Systems*, page e202501381, April 2026. 122

[22] Fred Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. 13(3):319–340, September 1989. 27, 176, 177

[23] Fred D. Davis. *A technology acceptance model for empirically testing new end-user information systems : theory and results*. Thesis, Massachusetts Institute of Technology, 1985. Accepted: 2005-08-08T16:11:32Z. viii, ix, 26, 27, 28, 90, 176, 178

[24] Zhamak Dehghani. Data Mesh Principles and Logical Architecture, December 2020. viii, 4, 5

[25] Zhamak Dehghani and Martin Fowler. *Data mesh: delivering data-driven value at scale*. Software architecture. O'Reilly, Beijing Boston Farnham Sebastopol Tokyo, first edition edition, 2022. 1, 4, 6

[26] Bernadette Dierckx de Casterlé, Chris Gastmans, Els Bryon, and Yvonne Denier. QUAGOL: A guide for qualitative data analysis. *International Journal of Nursing Studies*, 49(3):360–371, March 2012. 20

[27] Alex J.A. Donkers, Dujuan Yang, and Nico H.G. Baken. Linked Data for Smart Homes: Comparing RDF and Labeled Property Graphs: 8th Linked Data in Architecture and Construction Workshop, LDAC 2020. *LDAC 2020 Linked Data in Architecture and Construction*, pages 23–36, 2020. 9

[28] Aline Dresch, Daniel Pacheco Lacerda, and José Antônio Valle Antunes. Design Science Research. In Aline Dresch, Daniel Pacheco Lacerda, and José Antônio Valle Antunes Jr, editors, *Design Science Research: A Method for Science and Technology Advancement*, pages 67–102. Springer International Publishing, Cham, 2015. 15, 16

[29] Lisa Ehrlinger, Johannes Schrott, Martin Melichar, Nicolas Kirchmayr, and Wolfram Wöß. Data Catalogs: A Systematic Literature Review and Guidelines to Implementation. In Gabriele Kotsis, A. Min Tjoa, Ismail Khalil, Bernhard Moser, Atif Mashkoor, Johannes Sametinger, Anna Fensel, Jorge Martinez-Gil, Lukas Fischer, Gerald Czech, Florian Sobieczky, and Sohail Khan, editors, *Database and Expert Systems Applications - DEXA 2021 Workshops*, pages 148–158, Cham, 2021. Springer International Publishing. 81

[30] Ilker Etikan, Sulaiman Abubakar Musa, and Rukayya Sunusi Alkassim. Comparison of Convenience Sampling and Purposive Sampling. *American Journal of Theoretical and Applied Statistics*, 5(1):1–4, December 2015. 18

[31] Siamak Farshidi, Amir Saberhabibi, Behbod Eskafi, Niloofar Nikfarjam, Sadegh Eskandari, Slinger Jansen, Michel Chaudron, and Bedir Tekinerdogan. Empirical Evaluation of AI-Assisted Software Package Selection: A Knowledge Graph Approach, August 2025. arXiv:2508.05693 [cs]. 28

[32] Muhammad Afif Fathullah, Anusuyah Subbarao, Abdulaziz Ahmad, and Saravanan Muthaiyah. Merging Qualitative Design Methods With Design Science. *International Journal of Qualitative Methods*, 24:16094069251337585, November 2025. 17

[33] Christina Feilmayr and Wolfram Wöß. An analysis of ontologies and their success factors for application to business. *Data & Knowledge Engineering*, 101:1–23, January 2016. 81

[34] Sam Fletcher and Md Zahidul Islam. Comparing sets of patterns with the Jaccard index. *Australasian Journal of Information Systems*, 22, March 2018. 25

[35] Aldo Gangemi and Valentina Presutti. Ontology Design Patterns. In *Handbook on Ontologies*, pages 221–243. May 2009. Journal Abbreviation: Handbook on Ontologies. x, 65

[36] Saurabh Garg. Data Mesh and Decentralized Agentic Systems: A Unified Architecture for Autonomous Enterprise Decision-Making. *Journal of Computational Analysis and Applications (JoCAAA)*, 34(11):49–65, November 2025. 12

[37] Gartner. Data Quality: Why It Matters and How to Achieve It. 1

[38] Ewout Gelling, George Fletcher, and Michael Schmidt. Bridging graph data models: RDF, RDF-star, and property graphs as directed acyclic graphs, 2023. Version Number: 1. 8

[39] Maryam Ghasemaghaei and Goran Calic. Assessing the impact of big data on firm innovation performance: Big data is not always better data. *Journal of Business Research*, 108:147–162, January 2020. 1

[40] Pouya Ghiasnezhad Omran, Kerry Taylor, Sergio Rodríguez Méndez, and Armin Haller. Learning SHACL shapes from knowledge graphs. *Semantic Web*, 14(1):101–121, November 2022. 60

[41] Paul Gill, Kate Stewart, Elizabeth Treasure, and Barbara Chadwick. Methods of data collection in qualitative research: Interviews and focus groups. *British dental journal*, 204:291–5, April 2008. 18

[42] Abel Goedegebuure, Indika Kumara, Stefan Driessen, Willem-Jan Van Den Heuvel, Geert Monsieur, Damian Andrew Tamburri, and Dario Di Nucci. Data Mesh: A Systematic Gray Literature Review. *ACM Computing Surveys*, 57(1):1–36, January 2025. ix, 4, 81, 82

[43] Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is an Ontology? In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 1–17. Springer, Berlin, Heidelberg, 2009. 9, 87

[44] Honghao Gui, Shuofei Qiao, Jintian Zhang, Hongbin Ye, Mengshu Sun, Lei Liang, Jeff Z. Pan, Huajun Chen, and Ningyu Zhang. InstructIE: A Bilingual Instruction-based Information Extraction Dataset, July 2024. arXiv:2305.11527 [cs]. 131

[45] Zhenbei Guo, Fuliang Li, Peng Zhang, Xingwei Wang, and Jiannong Cao. NetKG: Synthesizing Interpretable Network Router Configurations With Knowledge Graph. *IEEE Transactions on Computers*, 74(11):3722–3735, November 2025. 12

[46] Ensar Hadziselimovic, Kaniz Fatema, Harshvardhan Pandit, and David Lewis. Linked Data Contracts to support Data Protection and Data Ethics in the Sharing of Scientific Data. 2017. 8

[47] Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. PiVe: Prompting with Iterative Verification Improving Graph-based Generative Capability of LLMs, May 2024. arXiv:2305.12392 [cs]. 10, 13

[48] Eberhard Hechler, Maryela Weihrauch, and Yan (Catherine) Wu. Data Fabric and Data Mesh Research Areas. In Eberhard Hechler, Maryela Weihrauch, and Yan (Catherine) Wu, editors, *Data Fabric and Data Mesh Approaches with AI: A Guide to AI-based Data Cataloging, Governance, Integration, Orchestration, and Consumption*, pages 375–392. Apress, Berkeley, CA, 2023. 54, 81, 123

[49] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research1. *Management Information Systems Quarterly*, 28(1):75–106, March 2004. viii, 15, 16, 23, 119, 128

[50] Jan Holmström, Mikko Ketokivi, and Ari-Pekka Hameri. Bridging Practice and Theory: A Design Science Approach. *Decision Sciences*, 40(1):65–87, 2009. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-5915.2008.00221.x. 16

[51] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. arXiv:2106.09685 [cs.CL]. 10

[52] Yuelin Hu, Futai Zou, Jiajia Han, Xin Sun, and Yilei Wang. LLM-TIKG: Threat intelligence knowledge graph construction utilizing large language model. *Computers & Security*, 145:103999, October 2024. 59

[53] Yujie Huang, Ran Yan, and Zhipeng Zhang. Automated knowledge extraction from marine accident reports using large language models: Graph construction and evaluation. *Ocean & Coastal Management*, 272:108015, January 2026. 13, 25, 26, 121

[54] Sungsoo Hwang. Utilizing Qualitative Data Analysis Software: A Review of Atlas.ti. *Social Science Computer Review*, 26(4):519–527, November 2008. 20

[55] Ana Iglesias-Molina, Dylan Van Assche, Julián Arenas-Guerrero, Ben De Meester, Christophe Debruyne, Samaneh Jozashoori, Pano Maria, Franck Michel, David Chaves-Fraga, and Anastasia Dimou. The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF. In Terry R. Payne, Valentina Presutti, Guilin Qi, María Poveda-Villalón, Giorgos Stoilos, Laura Hollink, Zoi Kaoudi, Gong Cheng, and Juanzi Li, editors, *The Semantic Web – ISWC 2023*, pages 152–175, Cham, 2023. Springer Nature Switzerland. 131

[56] Dama International. *DAMA-DMBOK: Data Management Body of Knowledge (2nd Edition)*. Technics Publications, LLC, Denville, NJ, USA, June 2017. 1, 6

[57] Paul Johannesson and Erik Perjons. *An Introduction to Design Science*. Springer Publishing Company, Incorporated, September 2014. 17, 21, 22

[58] Andrew Jones and Kevin Hu. *Driving data quality with data contracts: a comprehensive guide to building reliable, trusted, and effective data platforms*. Packt Publishing, Place of publication not identified, 2023. 1, 2

[59] Jaehun Jung, Jinhong Jung, and U Kang. Learning to Walk across Time for Interpretable Temporal Knowledge Graph Completion. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 786–795, New York, NY, USA, August 2021. Association for Computing Machinery. 12

[60] Krishna Kanagarla. DATA MESH DECENTRALISED DATA MANAGEMENT. *International Journal of Computer Networks and Wireless Communications*, January 2024. 1, 6, 16

[61] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. arXiv:2001.08361 [cs.LG]. 10

[62] Mayank Kejriwal. *Domain-Specific Knowledge Graph Construction*. SpringerBriefs in Computer Science. Springer International Publishing, Cham, 2019. 8, 9

[63] Judy Kendall. Axial Coding and the Grounded Theory Controversy. *Western Journal of Nursing Research*, 21(6):743–757, December 1999. 21

[64] Vijay Khatri and Carol V. Brown. Designing data governance. *Commun. ACM*, 53(1):148–152, January 2010. viii, 1, 6

[65] Shahrzad Khayatbashi, Sebastián Ferrada, and Olaf Hartig. Converting property graphs to RDF: a preliminary study of the practical impact of different mappings. In *Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, pages 1–9, Philadelphia Pennsylvania, June 2022. ACM. 60

[66] Norbert Koppenhagen, Oliver Gaß, and Benjamin Müller. Design Science Research in Action - Anatomy of Success Critical Activities for Rigor and Relevance. 2012. 17, 21, 22, 23

[67] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. Data Mesh: Concepts and Principles of a Paradigm Shift in Data Architectures. *Procedia Computer Science*, 196:263–271, January 2022. 4

[68] Steve Mann. Research Interviews: Modes and Types. In Steve Mann, editor, *The Research Interview: Reflective Practice and Reflexivity in Research Processes*, pages 86–113. Palgrave Macmillan UK, London, 2016. 17, 18

[69] Sara Marcucci, Natalia González Alarcón, Stefaan G. Verhulst, and Elena Wüllhorst. Informing the Global Data Future: Benchmarking Data Governance Frameworks. *Data & Policy*, 5:e30, January 2023. 6

[70] Ioana Ramona Martin, Tudor Cioara, Ionut Anghel, and Gabriel Arcas. Performance Evaluation of LLMs in Automated RDF Knowledge Graph Generation, February 2026. arXiv:2603.29878 [cs]. 79, 80, 120, 121

[71] Glaice Kelly Q. Monfardini, Jordana S. Salamon, and Monalessa P. Barcellos. Use of Competency Questions in Ontology Engineering: A Survey. In João Paulo A. Almeida, José Borbinha, Giancarlo Guizzardi, Sebastian Link, and Jelena Zdravkovic, editors, *Conceptual Modeling*, pages 45–64, Cham, 2023. Springer Nature Switzerland. 60

[72] Sergi Nadal, Petar Jovanovic, Besim Bilalli, and Oscar Romero. Operationalizing and automating Data Governance. *Journal of Big Data*, 9(1):117, December 2022. 6

[73] Sarah Oppold, Manuel Fritz, and Lucas Woltmann. Data Contracts to Leverage (De-)centralized Data Management in Manufacturing Industries: An Experience Report. 2025. 2, 6, 7

[74] Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. An Empirical Study of the Non-Determinism of ChatGPT in Code Generation. *ACM Trans. Softw. Eng. Methodol.*, 34(2):42:1–42:28, January 2025. 25, 78, 112

[75] Clare O'Callaghan, Justin Dwyer, and Penelope Schofield. Thematic analysis informed by grounded theory (TAG) in healthcare research: foundations and applications. *Qualitative Research in Psychology*, 21(3):279–306, July 2024. _eprint: https://doi.org/10.1080/14780887.2024.2347580. 17, 19, 20, 117

[76] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, July 2024. 13

[77] Tarun Parmar. Implementing CI/CD in Data Engineering: Streamlining Data Pipelines for Reliable and Scalable Solutions. January 2025. 83

[78] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, December 2016. 24, 59, 88

[79] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77, December 2007. _eprint: https://doi.org/10.2753/MIS0742-1222240302. viii, 15, 16, 23

[80] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web & Information Systems*, 10(2):7–34, April 2014. 87, 174

[81] Kiran Prakash. Designing data products, December 2024. 84

[82] Sumit Purohit, Nhuy Van, and George Chin. Semantic Property Graph for Scalable Knowledge Graph Analytics. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2672–2677, December 2021. 60, 131

[83] Kashif Rabbani, Matteo Lissandrini, Angela Bonifati, and Katja Hose. Transforming RDF Graphs to Property Graphs using Standardized Schemas. *Proceedings of the ACM on Management of Data*, 2(6):1–25, December 2024. 60, 131

[84] George F. Reed, Freyja Lynn, and Bruce D. Meade. Use of Coefficient of Variation in Assessing Variability of Quantitative Assays. *Clinical and Vaccine Immunology*, 9(6):1235–1239, November 2002. 25

[85] David Jebasingh S. Data Lakes and Data Mesh Architectures: Enabling Scalable and Decentralized Data Governance. *International Journal of Emerging Trends in Computer Science and Information Technology*, 5(4):16–22, October 2024. 6

[86] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications, 2024. Version Number: 2. 80

[87] Anselm Strauss and Juliet M. Corbin. *Basics of qualitative research: Grounded theory procedures and techniques*. Basics of qualitative research: Grounded theory procedures and techniques. Sage Publications, Inc, Thousand Oaks, CA, US, 1990. Pages: 270. viii, 21, 29, 30

[88] Heiner Stuckenschmidt and Michel Klein. Reasoning and change management in modular ontologies. *Data & Knowledge Engineering*, 63(2):200–223, November 2007. 64

[89] Devi Sunuwar and Monika Singh. Comparative Analysis of Relational and Graph Databases for Data Provenance: Performance, Queries, and Security Considerations. In *2023 World Conference on Communication & Computing (WCONF)*, pages 1–7, July 2023. 59

[90] Gerald I. Susman and Roger D. Evered. An Assessment of the Scientific Merits of Action Research. *Administrative Science Quarterly*, 23(4):582, December 1978. 15

[91] Mari Carmen Suárez-Figueroa. *NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. phd, Facultad de Informática (UPM), June 2010. 60

[92] Gytė Tamašauskaitė and Paul Groth. Defining a Knowledge Graph Development Process Through a Systematic Review. *ACM Trans. Softw. Eng. Methodol.*, 32(1):27:1–27:40, February 2023. 8

[93] Durairaj Thenmozhi, Makesh Vaibhav AG, and Naren Srinivasan M. Hierarchical knowledge graph based legal information retrieval from multiple documents using intelligent agents. *Artificial Intelligence and Law*, November 2025. 12

[94] Sushmi Thushara Sukumar, Chung-Horng Lung, Marzia Zaman, and Ritesh Panday. Knowledge Graph Generation and Application for Unstructured Data Using Data Processing Pipeline. *IEEE Access*, 12:136759–136770, 2024. 8, 10

[95] Yuanyuan Tian. The World of Graph Databases from An Industry Perspective. *SIGMOD Rec.*, 51(4):60–67, January 2023. 9

[96] Daniel Turner. Qualitative Interview Design: A Practical Guide for Novice Investigators. *The Qualitative Report*, 15(3):754–760, May 2010. 18

[97] Daniel van der Werf, João Moreira, and Jean Paul Sebastian Piest. Towards a Data Mesh Reference Architecture. In Monika Kaczmarek-Heß, Kristina Rosenthal, Marek Suchánek, Miguel Mira Da Silva, Henderik A. Proper, and Marianne Schnellmann, editors, *Enterprise Design, Operations, and Computing. EDOC 2024 Workshops*, pages 339–353, Cham, 2025. Springer Nature Switzerland. 7, 83

[98] Tom Van Eijk, Indika Kumara, Dario Di Nucci, Damian Andrew Tamburri, and Willem-Jan Van den Heuvel. Architectural Design Decisions for Self-Serve Data Platforms in Data Meshes. In *2024 IEEE 21st International Conference on Software Architecture (ICSA)*, pages 135–145, June 2024. ISSN: 2835-7043. 81

[99] Jerry J. Vaske, Jay Beaman, and Carly C. Sponarski. Rethinking Internal Consistency in Cronbach's Alpha. *Leisure Sciences*, 39(2):163–173, March 2017. _eprint: https://doi.org/10.1080/01490400.2015.1127189. 113, 114, 126

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. arXiv:1706.03762 [cs.CL]. viii, 10, 11

[101] Jan vom Brocke, Alan Hevner, and Alexander Maedche. Introduction to Design Science Research. In Jan vom Brocke, Alan Hevner, and Alexander Maedche, editors, *Design Science Research. Cases*, pages 1–13. Springer International Publishing, Cham, 2020. 16

[102] Robert A. Wagner and Michael J. Fischer. The String-to-String Correction Problem. *Journal of the ACM (JACM)*, 21(1):168–173, January 1974. 24

[103] Jeroen Wasser, Indika Kumara, Geert Monsieur, Willem-Jan Van Den Heuvel, and Damian Andrew Tamburri. Data Contracts in Data Mesh: A Systematic Gray Literature Review. In Boris Shishkov, editor, *Business Modeling and Software Design*, pages 21–38, Cham, 2025. Springer Nature Switzerland. 1, 2, 6, 7, 8, 123

[104] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2022. 10, 80

[105] Arif Wider, Simon Harrer, and Linus W. Dietz. AI-Assisted Data Governance with Data Mesh Manager. In *2025 IEEE International Conference on Web Services (ICWS)*, pages 963–965, July 2025. ISSN: 2836-3868. 12, 124

[106] Arif Wider, Sumedha Verma, and Atif Akhtar. Decentralized Data Governance as Part of a Data Mesh Platform: Concepts and Approaches. In *2023 IEEE International Conference on Web Services (ICWS)*, pages 746–754, July 2023. ISSN: 2836-3868. ix, 12, 81, 82, 83

[107] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, Berlin, Heidelberg, 2014. 17, 21, 22, 23

[108] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. Virtual Knowledge Graphs: An Overview of Systems and Use Cases. *Data Intelligence*, 1(3):201–223, June 2019. 131

[109] Lan Zhao, Yajuan Sun, Junhao Ren, Honglin Gao, and Gaoxi Xiao. Construction of waste-to-resource knowledge graph for industrial symbiosis identification using large language models. *Nature Communications*, 17(1):26, December 2025. 13

[110] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A Comprehensive Survey on Automatic Knowledge Graph Construction. *ACM Comput. Surv.*, 56(4):94:1–94:62, November 2023. 8, 9

[111] Xiangru Zhu, Zhixu Li, Xiaodan Wang, Xueyao Jiang, Penglei Sun, Xuwu Wang, Yanghua Xiao, and Nicholas Jing Yuan. Multi-Modal Knowledge Graph Construction and Application: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(2):715–735, February 2024. 10

Appendix A

Interview Questions

Table A.1: Exploratory Interview Questions and Rationales (Parts 1 & 2)

Question	Rationale
Part 1: Introduction *Goal: Validate the participant's expertise and contextualize their answers against their industry background.*	
Could you describe your current role, the company you work for, and its primary business domain?	Categorize the participant. Responses can differ between domains, and this helps analyze specific industry solutions.
How many years of experience or involvement do you have working with data architectures, specifically regarding data governance?	Validate the participant's expertise and knowledge.
Part 2: Architectural Context *Goal: Understand the friction between governance and autonomy that necessitates data contracts.*	
What is the most used type of data architecture within your job?	Segments the participants between distributed and centralized data architectures. Can help in determining whether challenges in data contracts are unique to either one of the architectures.
How do you balance the tension between domain autonomy and centralized governance standards?	Validates whether data contracts are essential to ensure governance in distributed architectures and by confirming the struggle with the specific trade-off between scalability and reliability.
When a data product is created or modified, what is the explicit process for establishing 'trust' with consumers before any data moves?	To investigate how trust works in distributed architectures.
Is there a self-serve data platform that automatically validates every product against global standards (e.g., schema validity) before it can be deployed?	Gather information about a state-of-the-art implementation of a self-serve data platform. Before we can improve the solution, we must document what automated validation currently exists.
What is your strategy for data discovery and metadata management to ensure data products are accessible and interpretable?	Validate the need for metadata requirements, as data contracts are as well for discoverability.

Table A.2: Exploratory Interview Questions and Rationales (Parts 3 & 4)

Question	Rationale
Part 3: Defining Data Contracts *Goal: Identify the "translation loss" between natural language (Business) and code (Technical)*	
What is the role and impact of data contracts within your data architecture?	Validates the necessity of data contracts.
Can you walk me through the process for defining a data contract?	To map the current workflow. This will lead to the understanding on how data contracts are currently defined.
How are Data Contracts currently specified/what language is used?	Gather information about current specifications/languages of data contracts.
Can you walk me through the journey of a data contract from its initial business definition to its final technical implementation?	Validate there is some disconnection between the business users and the technical users.
To what extent are your current agreements "machine-enforceable" versus "human-readable" (Word/Docs)?	See how current data contracts are implemented (YAML/JSON vs. Word/Excel) and whether there is a standardization.
Is there a trade-off you currently have to make between the two?	Confirm the problem, whether there is a trade-off between (semi-)structured and unstructured formats.
Part 4: Data Contract Enforcement *Goal: Quantify the manual burden and the cost of errors.*	
Describe the workflow required to update enforcement code when a business rule changes.	Looks at the labor (manual vs. automated) that is needed to update enforcement code.
How much of this process is manual versus automated, and what is the time-to-production?	Obtain quantifiable metrics. This could be the baseline on which the solution can be tested on.
When data enforcement issues arise, are they typically caught by mechanisms before consumption, or are they discovered by consumers downstream?	Assess operational risk and validate lack of reliability (poor data quality).
What are the root causes of these "slips"?	Gather information about system design and identify root causes, which can guide the solution.
How do you currently ensure consistency in policy enforcement across different technology stacks?	Confirms that a diverse technology stack leads to inconsistent governance.

Table A.3: Exploratory Interview Questions and Rationales (Parts 5 & 6)

Question	Rationale
Part 5: Challenges & Solution *Goal: Gather requirements for the artifact and identify barriers to AI adoption.*	
If you could design the ideal 'interface' for defining these contracts that satisfies both business and technical needs, what would it look like?	Gather requirements for proposing a solution.
What would the requirements and expectations be for a language specifying these contracts?	Gather requirements for proposing a solution.
What are the biggest barriers, technical, cultural, or trust-related, preventing your organization from automating the translation of requirements into code?	Gather information about the barrier to use and implement AI.
Part 6: Closing *Goal: Guided emergence and recruitment for the next phase.*	
Do you have any additional frustrations regarding data contract enforcement that we haven't discussed?	Discover any factors that might help guide the solution.
Would you be willing to participate in a follow-up interview/questionnaire about the artifact in a second stage of this research?	Secure participants for a follow-up phase.

Appendix B

Codes, Categories, & Themes

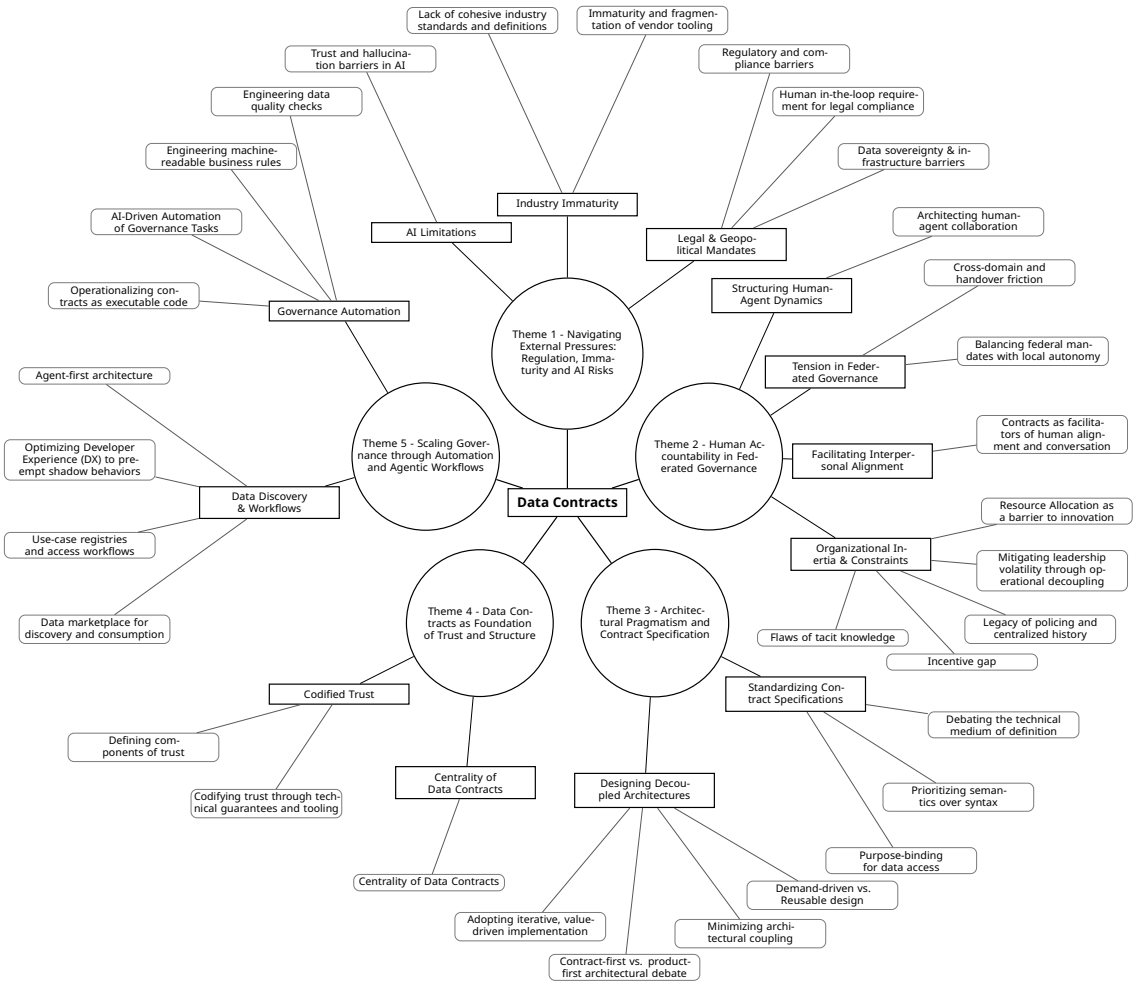

Figure B.1: Thematic map on Data Contracts.

Table B.1: Thematic Analysis: Hierarchy of Themes, Categories, and Codes

Category	Associated Codes
Theme 1 - Navigating External Pressures: Regulation, Immaturity and AI Risks	
Legal & Geopolitical Mandates	• Data sovereignty & infrastructure barriers • Human in-the-loop requirement for legal compliance • Regulatory and compliance barriers
Industry Immaturity	• Immaturity and fragmentation of vendor tooling • Lack of cohesive industry standards and definitions
AI Trust and Limitations	• Trust and hallucination barriers in AI
Theme 2 - Human Accountability in Federated Governance	
Facilitating Interpersonal Alignment	• Contracts as facilitators of human alignment and conversation
Organizational Inertia & Constraints	• Flaws of tacit knowledge • Incentive gap • Legacy of policing and centralized history • Mitigating leadership volatility through operational decoupling • Resource Allocation as a barrier to innovation
Tension in Federated Governance	• Balancing federal mandates with local autonomy • Cross-domain and handover friction
Structuring Human-Agent Dynamics	• Architecting human-agent collaboration
Theme 3 - Architectural Pragmatism and Contract Specification	
Designing Decoupled Architectures	• Demand-driven vs. Reusable design • Minimizing architectural coupling • Contract-first vs. product-first architectural debate • Adopting iterative, value-driven implementation
Standardizing Contract Specifications	• Debating the technical medium of definition • Prioritizing semantics over syntax • Purpose-binding for data access
Theme 4 - Data Contracts as Foundation of Trust and Structure	
Centrality of Data Contracts	• Centrality of Data Contracts
Codified Trust	• Defining components of trust • Codifying trust through technical guarantees and tooling
Theme 5 - Scaling Governance through Automation and Agentic Workflows	
Governance Automation	• Engineering data quality checks • Engineering machine-readable business rules • AI-Driven Automation of Governance Tasks • Operationalizing contracts as executable code
Data Discovery & Workflows	• Agent-first architecture • Optimizing Developer Experience (DX) to preempt shadow behaviors • Use-case registries and access workflows • Data marketplace for discovery and consumption

Appendix C

JSON Schema Definitions

The constraint-prompting approach described in Section 5.3.1.2 relies on Pydantic `BaseModel` classes that compile to a formal JSON Schema, against which the language model output is validated. Listings C.1 and C.2 show the resulting schemas.

```
{
  "$defs": {
    "Entity": {
      "properties": {
        "id": {
          "title": "Id",
          "type": "string"
        },
        "types": {
          "default": [],
          "items": {
            "type": "string"
          },
          "title": "Types",
          "type": "array"
        },
        "properties": {
          "type": "object",
          "additionalProperties": {
            "type": "string"
          },
          "default": {},
          "title": "Properties"
        }
      },
      "required": [
        "id"
      ],
      "title": "Entity",
      "type": "object"
    }
  },
  "properties": {
    "entities": {
      "items": {
        "$ref": "#/$defs/Entity"
      },
      "title": "Entities",
      "type": "array"
    }
  },
  "required": [
    "entities"
  ],
  "title": "EntityExtractionResult",
  "type": "object"
}
```

Listing C.1: JSON Schema for the entity extraction result.

```json
{
  "$defs": {
    "Relationship": {
      "properties": {
        "subject": {
          "title": "Subject",
          "type": "string"
        },
        "predicate": {
          "title": "Predicate",
          "type": "string"
        },
        "object": {
          "title": "Object",
          "type": "string"
        }
      },
      "required": [
        "subject",
        "predicate",
        "object"
      ],
      "title": "Relationship",
      "type": "object"
    }
  },
  "properties": {
    "relationships": {
      "items": {
        "$ref": "#/$defs/Relationship"
      },
      "title": "Relationships",
      "type": "array"
    }
  },
  "required": [
    "relationships"
  ],
  "title": "RelationshipExtractionResult",
  "type": "object"
}
```

Listing C.2: JSON Schema for the relationship extraction result.

Appendix D

Prompts

In the following prompts used for knowledge graph generation, a standard set of prefixes was provided to the model. To reduce redundancy in this appendix, the **Allowed Prefixes** section included in every prompt is listed once below:

```
--- Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dprod: <https://ekgf.github.io/dprod/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix csvw: <http://www.w3.org/ns/csvw#> .
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .
@prefix org: <http://www.w3.org/ns/org#> .
@prefix dqv: <http://www.w3.org/ns/dqv#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix iso: <https://iso25000.com/index.php/en/iso-25000-standards/iso-25012#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dpv: <http://www.w3.org/ns/dpv#> .
@prefix dcont: <http://thesis.tdhoven.nl/ontologies/datacontract#> .
@prefix dme: <http://thesis.tdhoven.nl/ontologies/data-mesh-sample-environment#> .
```

D.1 Prompt Strategies Examples

The one-shot and OS-CoT prompt strategies both embed the same worked demonstrations alongside their respective task descriptions. To avoid duplicating these blocks across both prompts, the three examples (one for entity extraction, one for relation extraction, and one for direct RDF extraction) are listed once below and referenced from each prompt where applicable.

D.1.1 Entity Extraction Example

```
INPUT TEXT:
"""
Existing Knowledge Graph context:

dme:ProductRecommendationEngine a dprod:DataProduct ;
   dct:identifier "product_recommendation_engine" ;
   rdfs:label "Product Recommendation Engine" ;
   dprod:domain "production" ;
   dprod:inputPort dme:ProductRecommendationEngineInputPort ;
   dprod:outputPort dme:ProductRecommendationEngineOutputPort .

dme:ProductRecommendationEngineOutputPort a dcat:DataService, odrl:Asset .

dme:ProductRecommendationEngineInputPort a dcat:DataService ;
   rdfs:label "Input Port consuming other data products" .

The data contract specification:

id: product_recommendation_engine
```

```
name: Product Recommendation Engine
status: active
domain: production

team:
  id: mp_team
  name: Manufacturing & Production Team
  members:
    - username: janedoe
      role: Data Product Owner

schema:
  - id: obj_product_recommendation_engine
    physicalName: product_recommendation_engine
    physicalType: view
    properties:
      - name: ProductID
        physicalType: integer
        primaryKey: true
        required: true
        quality:
          - id: product_id_completeness
            metric: nullValues
            mustBe: 0
            unit: rows
      - name: StandardCost
        physicalType: numeric(19,4)
        required: true
        quality:
          - type: sql
            query: SELECT COUNT(*) FROM {table} WHERE {column} IS NOT NULL
            mustBeGreaterOrEqualTo: 0.0000

slaProperties:
  - id: product_recommendation_engine_availability
    property: availability
    value: 99.9%
  - id: product_recommendation_engine_freshness
    property: latency
    value: 24 hours

roles:
  - id: sales_rep_read_access
    role: sales_representative
    access: read
"""

OUTPUT JSON:
"""
{
  "entities": [
    {
      "id": "dme:ProductRecommendationEngine",
      "types": ["dprod:DataProduct"],
      "properties": {
        "dct:identifier": "9d8e7f6a-5b4c-3d2e-1f0a-ProductRecommendationEngine",
        "dprod:purpose": "Recommendation engine for customers.",
        "dcat:keyword": "product_recommendation_engine"
      }
    },
    {
      "id": "dme:ProductRecommendationEngineOutputPort",
      "types": ["dcat:DataService", "odrl:Asset"],
      "properties": {
        "rdfs:label": "postgres-development",
        "dprod:protocol": "postgresql",
        "dcat:endpointURL": "postgresql://postgres-development:5432/thesis_db"
      }
    },
    {
      "id": "dme:ProductRecommendationEngineDataset",
      "types": ["dcat:Dataset"],
      "properties": {
        "dct:description": "Flattened product view."
      }
    },
    {
      "id": "dme:ProductRecommendationEngineSchema",
      "types": ["csvw:Schema"],
      "properties": {
```

```
      "csvw:primaryKey": "ProductID"
    }
  },
  {
    "id": "dme:ProductRecommendationEngineProductID",
    "types": ["csvw:Column", "odrl:Asset"],
    "properties": {
      "csvw:name": "ProductID",
      "csvw:datatype": "integer",
      "csvw:required": true
    }
  },
  {
    "id": "dme:ProductRecommendationEngineStandardCost",
    "types": ["csvw:Column", "odrl:Asset"],
    "properties": {
      "csvw:name": "StandardCost",
      "csvw:datatype": "numeric(19,4)",
      "csvw:required": true
    }
  },
  {
    "id": "dme:MPTeam",
    "types": ["org:OrganizationalUnit", "odrl:Party"],
    "properties": {
      "foaf:name": "Manufacturing Team"
    }
  },
  {
    "id": "dme:JohnDoe",
    "types": ["foaf:Person", "prov:Person", "odrl:Party"],
    "properties": {
      "foaf:givenName": "John Doe",
      "dct:identifier": "johndoe"
    }
  },
  {
    "id": "dme:MembershipJohnDoe",
    "types": ["org:Membership"],
    "properties": {
      "prov:startedAtTime": "2025-08-05",
      "prov:endedAtTime": "2026-01-31",
      "org:role": "Data Product Owner"
    }
  },
  {
    "id": "dme:OwnershipJohnDoeProductRecommendationEngine",
    "types": ["dcont:ProductOwnership"],
    "properties": {
      "prov:startedAtTime": "2025-08-05",
      "prov:endedAtTime": "2026-01-31",
      "dct:description": "Past owner."
    }
  },
  {
    "id": "dme:JaneDoe",
    "types": ["foaf:Person", "prov:Person", "odrl:Party"],
    "properties": {
      "foaf:givenName": "Jane Doe",
      "dct:identifier": "janedoe"
    }
  },
  {
    "id": "dme:MembershipJaneDoe",
    "types": ["org:Membership"],
    "properties": {
      "org:role": "Data Product Owner"
    }
  },
  {
    "id": "dme:OwnershipJaneDoeProductRecommendationEngine",
    "types": ["dcont:ProductOwnership"],
    "properties": {
      "prov:startedAtTime": "2026-02-01"
    }
  },
  {
    "id": "dme:ProductRecommendationEngineDutyProductIDCompleteness",
    "types": ["odrl:Duty"],
    "properties": {}
```

```
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintProductIDCompleteness",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "eq",
    "odrl:rightOperand": 0,
    "odrl:unit": "rows"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineDutyProductIDUniqueness",
   "types": ["odrl:Duty"],
   "properties": {}
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintProductIDUniqueness",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "lt",
    "odrl:rightOperand": 1,
    "odrl:unit": "percent"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineDutyStandardCostQuality",
   "types": ["odrl:Duty"],
   "properties": {
    "dct:description": "SQL Rule: SELECT COUNT(*) FROM {table} WHERE {column} IS NOT NULL"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintStandardCostMin",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "gteq",
    "odrl:rightOperand": 0,
    "odrl:unit": "euros"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintStandardCostMax",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "lteq",
    "odrl:rightOperand": 10000,
    "odrl:unit": "euros"
   }
  },
  {
   "id": "dme:PolicyProductRecommendationEngine",
   "types": ["dcont:DataContractPolicy"],
   "properties": {
    "dct:identifier": "product_recommendation_engine_contract"
   }
  },
  {
   "id": "dme:SalesRepresentatives",
   "types": ["odrl:Party"],
   "properties": {
    "foaf:name": "sales_representative"
   }
  },
  {
   "id": "dme:ProductRecommendationEnginePermissionUsageRead",
   "types": ["odrl:Rule"],
   "properties": {
    "dct:description": "Usage: Join on 'ProductID'. Do not use for historical analysis."
   }
  },
  {
   "id": "dme:ProductRecommendationEngineProhibitionLimitations",
   "types": ["odrl:Rule"],
   "properties": {
    "dct:description": "Limitations: Cost data is confidential. Do not expose externally."
   }
  },
  {
   "id": "dme:ProductRecommendationEngineDutyAvailability",
   "types": ["odrl:Duty"],
```

```
    "properties": {}
   },
   {
    "id": "dme:ProductRecommendationEngineConstraintAvailability",
    "types": ["odrl:Constraint"],
    "properties": {
     "odrl:operator": "gteq",
     "odrl:rightOperand": 99.9,
     "odrl:unit": "percentage"
    }
   },
   {
    "id": "dme:ProductRecommendationEngineDutySLOFreshness",
    "types": ["odrl:Duty"],
    "properties": {}
   },
   {
    "id": "dme:ProductRecommendationEngineConstraintSLOFreshness",
    "types": ["odrl:Constraint"],
    "properties": {
     "odrl:operator": "lteq",
     "odrl:rightOperand": 5,
     "odrl:unit": "minutes"
    }
   }
  ]
}
"""
```

D.1.2 Relation Extraction Example

```
--- Example ---
INPUT TEXT:
"""
Existing Knowledge Graph context:

dme:ProductRecommendationEngine a dprod:DataProduct ;
   dct:identifier "product_recommendation_engine" ;
   rdfs:label "Product Recommendation Engine" ;
   dprod:domain "production" ;
   dprod:inputPort dme:ProductRecommendationEngineInputPort ;
   dprod:outputPort dme:ProductRecommendationEngineOutputPort .

dme:ProductRecommendationEngineOutputPort a dcat:DataService, odrl:Asset .

dme:ProductRecommendationEngineInputPort a dcat:DataService ;
   rdfs:label "Input Port consuming other data products" .

The data contract specification:

id: product_recommendation_engine
name: Product Recommendation Engine
status: active
domain: production

team:
  id: mp_team
  name: Manufacturing & Production Team
  members:
    - username: janedoe
      role: Data Product Owner

schema:
  - id: obj_product_recommendation_engine
    physicalName: product_recommendation_engine
    physicalType: view
    properties:
      - name: ProductID
        physicalType: integer
        primaryKey: true
        required: true
        quality:
          - id: product_id_completeness
            metric: nullValues
            mustBe: 0
            unit: rows
```

```
    - name: StandardCost
      physicalType: numeric(19,4)
      required: true
      quality:
        - type: sql
          query: SELECT COUNT(*) FROM {table} WHERE {column} IS NOT NULL
          mustBeGreaterOrEqualTo: 0.0000

slaProperties:
  - id: product_recommendation_engine_availability
    property: availability
    value: 99.9%
  - id: product_recommendation_engine_freshness
    property: latency
    value: 24 hours

roles:
  - id: sales_rep_read_access
    role: sales_representative
    access: read
"""

PROVIDED ENTITIES:
"{
  "entities": [
    {
      "id": "dme:ProductRecommendationEngine",
      "types": ["dprod:DataProduct"],
      "properties": {
        "dct:identifier": "product_recommendation_engine",
        "dcat:keyword": "product_recommendation_engine"
      }
    },
    {
      "id": "dme:ProductRecommendationEngineOutputPort",
      "types": ["dcat:DataService", "odrl:Asset"],
      "properties": {
        "rdfs:label": "postgres-development",
        "dprod:protocol": "postgresql",
        "dcat:endpointURL": "postgresql://postgres-development:5432/thesis_db"
      }
    },
    {
      "id": "dme:ProductRecommendationEngineDataset",
      "types": ["dcat:Dataset"],
      "properties": {
        "dct:description": "Flattened product view."
      }
    },
    {
      "id": "dme:ProductRecommendationEngineSchema",
      "types": ["csvw:Schema"],
      "properties": {
        "csvw:primaryKey": "ProductID"
      }
    },
    {
      "id": "dme:ProductRecommendationEngineProductID",
      "types": ["csvw:Column", "odrl:Asset"],
      "properties": {
        "csvw:name": "ProductID",
        "csvw:datatype": "integer",
        "csvw:required": true
      }
    },
    {
      "id": "dme:ProductRecommendationEngineStandardCost",
      "types": ["csvw:Column", "odrl:Asset"],
      "properties": {
        "csvw:name": "StandardCost",
        "csvw:datatype": "numeric(19,4)",
        "csvw:required": true
      }
    },
    {
      "id": "dme:MPTeam",
      "types": ["org:OrganizationalUnit", "odrl:Party"],
      "properties": {
        "foaf:name": "Manufacturing Team"
      }
```

```
  },
  {
   "id": "dme:JaneDoe",
   "types": ["foaf:Person", "prov:Person", "odrl:Party"],
   "properties": {
    "foaf:givenName": "Jane Doe",
    "dct:identifier": "janedoe"
   }
  },
  {
   "id": "dme:MembershipJaneDoe",
   "types": ["org:Membership"],
   "properties": {
    "org:role": "Data Product Owner"
   }
  },
  {
   "id": "dme:OwnershipJaneDoeProductRecommendationEngine",
   "types": ["dcont:ProductOwnership"],
   "properties": {
   }
  },
  {
   "id": "dme:ProductRecommendationEngineDutyProductIDCompleteness",
   "types": ["odrl:Duty"],
   "properties": {}
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintProductIDCompleteness",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "eq",
    "odrl:rightOperand": 0,
    "odrl:unit": "rows"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineDutyProductIDUniqueness",
   "types": ["odrl:Duty"],
   "properties": {}
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintProductIDUniqueness",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "lt",
    "odrl:rightOperand": 1,
    "odrl:unit": "percent"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineDutyStandardCostQuality",
   "types": ["odrl:Duty"],
   "properties": {
    "dct:description": "SQL Rule: SELECT COUNT(*) FROM {table} WHERE {column} IS NOT NULL"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintStandardCostMin",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "gteq",
    "odrl:rightOperand": 0,
    "odrl:unit": "euros"
   }
  },
  {
   "id": "dme:ProductRecommendationEngineConstraintStandardCostMax",
   "types": ["odrl:Constraint"],
   "properties": {
    "odrl:operator": "lteq",
    "odrl:rightOperand": 10000,
    "odrl:unit": "euros"
   }
  },
  {
   "id": "dme:PolicyProductRecommendationEngine",
   "types": ["dcont:DataContractPolicy"],
   "properties": {
    "dct:identifier": "product_recommendation_engine_contract"
```

```
    }
   },
   {
    "id": "dme:SalesRepresentatives",
    "types": ["odrl:Party"],
    "properties": {
     "foaf:name": "sales_representative"
    }
   },
   {
    "id": "dme:ProductRecommendationEnginePermissionUsageRead",
    "types": ["odrl:Rule"],
    "properties": {
     "dct:description": "Usage: Join on 'ProductID'. Do not use for historical analysis."
    }
   },
   {
    "id": "dme:ProductRecommendationEngineProhibitionLimitations",
    "types": ["odrl:Rule"],
    "properties": {
     "dct:description": "Limitations: Cost data is confidential. Do not expose externally."
    }
   },
   {
    "id": "dme:ProductRecommendationEngineDutyAvailability",
    "types": ["odrl:Duty"],
    "properties": {}
   },
   {
    "id": "dme:ProductRecommendationEngineConstraintAvailability",
    "types": ["odrl:Constraint"],
    "properties": {
     "odrl:operator": "gteq",
     "odrl:rightOperand": 99.9,
     "odrl:unit": "percentage"
    }
   },
   {
    "id": "dme:ProductRecommendationEngineDutySLOFreshness",
    "types": ["odrl:Duty"],
    "properties": {}
   },
   {
    "id": "dme:ProductRecommendationEngineConstraintSLOFreshness",
    "types": ["odrl:Constraint"],
    "properties": {
     "odrl:operator": "lteq",
     "odrl:rightOperand": 5,
     "odrl:unit": "minutes"
    }
   }
  ]
}"

OUTPUT JSON:
{
  "relationships": [
   { "subject": "dme:ProductRecommendationEngine", "predicate": "dprod:dataProductOwner", "object":
        "dme:JaneDoe" },
   { "subject": "dme:ProductRecommendationEngine", "predicate": "dprod:outputDataset", "object": "
       dme:ProductRecommendationEngineDataset" },
   { "subject": "dme:ProductRecommendationEngineDataset", "predicate": "dct:conformsTo", "object":
        "dme:ProductRecommendationEngineSchema" },
   { "subject": "dme:ProductRecommendationEngineSchema", "predicate": "csvw:column", "object": "
       dme:ProductRecommendationEngineProductID" },
   { "subject": "dme:ProductRecommendationEngineSchema", "predicate": "csvw:column", "object": "
       dme:ProductRecommendationEngineStandardCost" },
   { "subject": "dme:MembershipJaneDoe", "predicate": "org:organization", "object": "dme:MPTeam" },

   { "subject": "dme:MembershipJaneDoe", "predicate": "org:member", "object": "dme:JaneDoe" },
   { "subject": "dme:OwnershipJaneDoeProductRecommendationEngine", "predicate": "dcont:dataProduct
       ", "object": "dme:ProductRecommendationEngine" },
   { "subject": "dme:OwnershipJaneDoeProductRecommendationEngine", "predicate": "dprod:
       dataProductOwner", "object": "dme:JaneDoe" },
   { "subject": "dme:ProductRecommendationEngineDutyProductIDCompleteness", "predicate": "odrl:
       target", "object": "dme:ProductRecommendationEngineProductID" },
   { "subject": "dme:ProductRecommendationEngineDutyProductIDCompleteness", "predicate": "odrl:
       constraint", "object": "dme:ProductRecommendationEngineConstraintProductIDCompleteness" },
   { "subject": "dme:ProductRecommendationEngineConstraintProductIDCompleteness", "predicate": "
       odrl:leftOperand", "object": "iso:Completeness" },
```

```
   { "subject": "dme:ProductRecommendationEngineDutyProductIDUniqueness", "predicate": "odrl:
        target", "object": "dme:ProductRecommendationEngineProductID" },
   { "subject": "dme:ProductRecommendationEngineDutyProductIDUniqueness", "predicate": "odrl:
        constraint", "object": "dme:ProductRecommendationEngineConstraintProductIDUniqueness" },
   { "subject": "dme:ProductRecommendationEngineConstraintProductIDUniqueness", "predicate": "odrl:
        leftOperand", "object": "iso:Consistency" },
   { "subject": "dme:ProductRecommendationEngineDutyStandardCostQuality", "predicate": "odrl:
        target", "object": "dme:ProductRecommendationEngineStandardCost" },
   { "subject": "dme:ProductRecommendationEngineDutyStandardCostQuality", "predicate": "odrl:
        constraint", "object": "dme:ProductRecommendationEngineConstraintStandardCostMin" },
   { "subject": "dme:ProductRecommendationEngineDutyStandardCostQuality", "predicate": "odrl:
        constraint", "object": "dme:ProductRecommendationEngineConstraintStandardCostMax" },
   { "subject": "dme:ProductRecommendationEngineConstraintStandardCostMin", "predicate": "odrl:
        leftOperand", "object": "iso:Compliance" },
   { "subject": "dme:ProductRecommendationEngineConstraintStandardCostMax", "predicate": "odrl:
        leftOperand", "object": "iso:Compliance" },
   { "subject": "dme:PolicyProductRecommendationEngine", "predicate": "odrl:target", "object": "
        dme:ProductRecommendationEngineOutputPort" },
   { "subject": "dme:PolicyProductRecommendationEngine", "predicate": "odrl:assigner", "object": "
        dme:MPTeam" },
   { "subject": "dme:PolicyProductRecommendationEngine", "predicate": "odrl:assignee", "object": "
        dme:SalesRepresentatives" },
   { "subject": "dme:PolicyProductRecommendationEngine", "predicate": "odrl:permission", "object":
         "dme:ProductRecommendationEnginePermissionUsageRead" },
   { "subject": "dme:PolicyProductRecommendationEngine", "predicate": "odrl:prohibition", "object":
         "dme:ProductRecommendationEngineProhibitionLimitations" },
   { "subject": "dme:PolicyProductRecommendationEngine", "predicate": "odrl:obligation", "object":
         "dme:ProductRecommendationEngineDutyAvailability" },
   { "subject": "dme:PolicyProductRecommendationEngine", "predicate": "odrl:obligation", "object":
         "dme:ProductRecommendationEngineDutySLOFreshness" },
   { "subject": "dme:ProductRecommendationEnginePermissionUsageRead", "predicate": "odrl:assignee",
         "object": "dme:SalesRepresentatives" },
   { "subject": "dme:ProductRecommendationEnginePermissionUsageRead", "predicate": "odrl:action",
        "object": "dcont:read" },
   { "subject": "dme:ProductRecommendationEngineProhibitionLimitations", "predicate": "odrl:action
        ", "object": "dcont:distribute" },
   { "subject": "dme:ProductRecommendationEngineDutyAvailability", "predicate": "odrl:action", "
        object": "dcont:ensure" },
   { "subject": "dme:ProductRecommendationEngineDutyAvailability", "predicate": "odrl:constraint",
         "object": "dme:ProductRecommendationEngineConstraintAvailability" },
   { "subject": "dme:ProductRecommendationEngineConstraintAvailability", "predicate": "odrl:
        leftOperand", "object": "iso:Availability" },
   { "subject": "dme:ProductRecommendationEngineDutySLOFreshness", "predicate": "odrl:action", "
        object": "dcont:ensure" },
   { "subject": "dme:ProductRecommendationEngineDutySLOFreshness", "predicate": "odrl:constraint",
         "object": "dme:ProductRecommendationEngineConstraintSLOFreshness" }
  ]
}
"""
```

D.1.3 Direct RDF Extraction Example

```
INPUT TEXT:
"""
This text is a data contract for an existing Data Product.
The following is already inside the knowledge graph:

dme:ProductRecommendationEngine a dprod:DataProduct ;
   dct:identifier "product_recommendation_engine" ;
   rdfs:label "Product Recommendation Engine" ;
   dprod:domain "production" ;
   dprod:inputPort dme:ProductRecommendationEngineInputPort ;
   dprod:outputPort dme:ProductRecommendationEngineOutputPort .

dme:ProductRecommendationEngineOutputPort a dcat:DataService, odrl:Asset .

dme:ProductRecommendationEngineInputPort a dcat:DataService ;
   rdfs:label "Input Port consuming other data products" .

The new data contract is:

version: 1.0.0
kind: DataContract
id: product_recommendation_engine
name: Product Recommendation Engine
dataProduct: product_recommendation_engine
```

```
description:
  purpose: Recommendation engine for customers.
  limitations: Cost data is confidential. Do not expose externally.
  usage: Join on 'ProductID'. Do not use for historical analysis.

tags:
  - product_recommendation_engine

servers:
  - server: postgres-development
    type: postgres
    host: 127.0.0.1
    port: 5432
    database: thesis_db

team:
  id: mp_team
  name: Manufacturing Team
  members:
    - username: johndoe
      name: John Doe
      role: Data Product Owner
      dateIn: "2025-08-05"
      dateOut: "2026-01-31"
      description: Past owner.
    - username: janedoe
      name: Jane Doe
      role: Data Product Owner
      dateIn: "2026-02-01"

schema:
  - id: obj_product_recommendation_engine
    physicalName: product_recommendation_engine
    description: Flattened product view.
    properties:
      - name: ProductID
        datatype: integer
        primaryKey: true
        required: true
        quality:
          - id: product_id_completeness
            metric: nullValues
            mustBe: 0
            unit: rows
            dimension: completeness
          - id: product_id_uniqueness
            metric: duplicateValues
            mustBeLessThan: 1
            unit: percent
            dimension: uniqueness
      - name: StandardCost
        datatype: numeric(19,4)
        required: true
        quality:
          - type: sql
            query: "SELECT COUNT(*) FROM {table} WHERE {column} IS NOT NULL"
            mustBeGreaterOrEqualTo: 0.0000
            mustBeLessThanOrEqualTo: 10000.0000

slaProperties:
  - id: product_recommendation_engine_availability
    property: availability
    value: 99.9%

roles:
  - id: sales_rep_read_access
    role: sales_representative
    access: read

customProperties:
  - id: slo_freshness_target
    property: slo_freshness_target
    value: "99% in 5 minutes"
"""

OUTPUT TTL:
"""
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dprod: <https://ekgf.github.io/dprod/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix csvw: <http://www.w3.org/ns/csvw#> .
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .
@prefix org: <http://www.w3.org/ns/org#> .
@prefix dqv: <http://www.w3.org/ns/dqv#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix iso: <https://iso25000.com/index.php/en/iso-25000-standards/iso-25012#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dpv: <http://www.w3.org/ns/dpv#> .
@prefix dcont: <http://thesis.tdhoven.nl/ontologies/datacontract#> .
@prefix dme: <http://thesis.tdhoven.nl/ontologies/data-mesh-sample-environment#> .

dme:ProductRecommendationEngine a dprod:DataProduct ;
   dct:identifier "9d8e7f6a-5b4c-3d2e-1f0a-ProductRecommendationEngine" ;
   dprod:purpose "Recommendation engine for customers." ;
   dcat:keyword "product_recommendation_engine" ;
   dprod:dataProductOwner dme:janedoe ;
   dprod:outputDataset dme:ProductRecommendationEngineDataset .

dme:ProductRecommendationEngineOutputPort a dcat:DataService, odrl:Asset ;
   rdfs:label "postgres-development" ;
   dprod:protocol "postgresql" ;
   dcat:endpointURL <postgresql://postgres-development:5432/thesis_db> .

dme:ProductRecommendationEngineDataset a dcat:Dataset ;
   dct:description "Flattened product view." ;
   dct:conformsTo dme:ProductRecommendationEngineSchema .

dme:ProductRecommendationEngineSchema a csvw:Schema ;
   csvw:primaryKey "ProductID" ;
   csvw:column dme:ProductRecommendationEngineProductID, dme:
        ProductRecommendationEngineStandardCost .

dme:ProductRecommendationEngineProductID a csvw:Column, odrl:Asset ;
   csvw:name "ProductID" ;
   csvw:datatype "integer" ;
   csvw:required true .

dme:ProductRecommendationEngineStandardCost a csvw:Column, odrl:Asset ;
   csvw:name "StandardCost" ;
   csvw:datatype "numeric(19,4)" ;
   csvw:required true .

dme:MPTeam a org:OrganizationalUnit, odrl:Party ;
   foaf:name "Manufacturing Team" .

dme:JohnDoe a foaf:Person, prov:Person, odrl:Party ;
   foaf:givenName "John Doe" ;
   dct:identifier "johndoe" .

dme:MembershipJohnDoe a org:Membership ;
   org:organization dme:MPTeam ;
   org:member dme:JohnDoe ;
   prov:startedAtTime "2025-08-05"^^xsd:date ;
   prov:endedAtTime "2026-01-31"^^xsd:date ;
   org:role "Data Product Owner" .

dme:OwnershipJohnDoeProductRecommendationEngine a dcont:ProductOwnership ;
   dcont:dataProduct dme:ProductRecommendationEngine ;
   dprod:dataProductOwner dme:JohnDoe ;
   prov:startedAtTime "2025-08-05"^^xsd:date ;
   prov:endedAtTime "2026-01-31"^^xsd:date ;
   dct:description "Past owner." .

dme:JaneDoe a foaf:Person, prov:Person, odrl:Party ;
   foaf:givenName "Jane Doe" ;
   dct:identifier "janedoe" .

dme:MembershipJaneDoe a org:Membership ;
   org:organization dme:MPTeam ;
   org:member dme:JaneDoe ;
   org:role "Data Product Owner" .

dme:OwnershipJaneDoeProductRecommendationEngine a dcont:ProductOwnership ;
   dcont:dataProduct dme:ProductRecommendationEngine ;
   dprod:dataProductOwner dme:JaneDoe ;
```

```
   prov:startedAtTime "2026-02-01"^^xsd:date .

dme:ProductRecommendationEngineDutyProductIDCompleteness a odrl:Duty ;
   odrl:target dme:ProductRecommendationEngineProductID ;
   odrl:constraint dme:ProductRecommendationEngineConstraintProductIDCompleteness .

dme:ProductRecommendationEngineConstraintProductIDCompleteness a odrl:Constraint ;
   odrl:leftOperand iso:Completeness ;
   odrl:operator "eq" ;
   odrl:rightOperand 0 ;
   odrl:unit "rows" .

dme:ProductRecommendationEngineDutyProductIDUniqueness a odrl:Duty ;
   odrl:target dme:ProductRecommendationEngineProductID ;
   odrl:constraint dme:ProductRecommendationEngineConstraintProductIDUniqueness .

dme:ProductRecommendationEngineConstraintProductIDUniqueness a odrl:Constraint ;
   odrl:leftOperand iso:Consistency ;
   odrl:operator "lt" ;
   odrl:rightOperand 1 ;
   odrl:unit "percentage" .

dme:ProductRecommendationEngineDutyStandardCostQuality a odrl:Duty ;
   odrl:target dme:ProductRecommendationEngineStandardCost ;
   dct:description "SQL Rule: SELECT COUNT(*) FROM {table} WHERE {column} IS NOT NULL" ;
   odrl:constraint dme:ProductRecommendationEngineConstraintStandardCostMin, dme:
        ProductRecommendationEngineConstraintStandardCostMax .

dme:ProductRecommendationEngineConstraintStandardCostMin a odrl:Constraint ;
   odrl:leftOperand iso:Compliance ;
   odrl:operator "gteq" ;
   odrl:rightOperand 0 ;
   odrl:unit "euros" .

dme:ProductRecommendationEngineConstraintStandardCostMax a odrl:Constraint ;
   odrl:leftOperand iso:Compliance ;
   odrl:operator "lteq" ;
   odrl:rightOperand 10000 ;
   odrl:unit "euros" .

dme:PolicyProductRecommendationEngine a dcont:DataContractPolicy ;
   dct:identifier "ProductRecommendationEngine_contract" ;
   odrl:target dme:ProductRecommendationEngineOutputPort ;
   odrl:assigner dme:MPTeam ;
   odrl:assignee dme:SalesRepresentatives ;
   odrl:permission dme:ProductRecommendationEnginePermissionUsageRead ;
   odrl:prohibition dme:ProductRecommendationEngineProhibitionLimitations ;
   odrl:obligation dme:ProductRecommendationEngineDutyAvailability, dme:
        ProductRecommendationEngineDutySLOFreshness .

dme:SalesRepresentatives a odrl:Party ;
   foaf:name "sales_representative" .

dme:ProductRecommendationEnginePermissionUsageRead a odrl:Rule ;
   odrl:assignee dme:SalesRepresentatives ;
   odrl:action dcont:read ;
   dct:description "Usage: Join on 'ProductID'. Do not use for historical analysis." .

dme:ProductRecommendationEngineProhibitionLimitations a odrl:Rule ;
   odrl:action dcont:distribute ;
   dct:description "Limitations: Cost data is confidential. Do not expose externally." .

dme:ProductRecommendationEngineDutyAvailability a odrl:Duty ;
   odrl:action dcont:ensure ;
   odrl:constraint dme:ProductRecommendationEngineConstraintAvailability .

dme:ProductRecommendationEngineConstraintAvailability a odrl:Constraint ;
   odrl:leftOperand dme:availability ;
   odrl:operator "gteq" ;
   odrl:rightOperand 99.9 ;
   odrl:unit "percentage" .

dme:ProductRecommendationEngineDutySLOFreshness a odrl:Duty ;
   odrl:action dcont:ensure ;
   odrl:constraint dme:ProductRecommendationEngineConstraintSLOFreshness .

dme:ProductRecommendationEngineConstraintSLOFreshness a odrl:Constraint ;
   odrl:leftOperand "slo_freshness_target" ;
   odrl:operator "lteq" ;
   odrl:rightOperand 5 ;
```

```
   odrl:unit "minutes" .
"""
```

D.2 Zero-Shot Prompt

D.2.1 Prompt 1: Entity Extraction

```
You are an expert knowledge graph architect extracting entities and their intrinsic properties
from data contract documents.

Your task is to identify key elements, assign their structural types, and extract their literal
properties into a strict JSON schema.

[... Allowed Prefixes see introduction of Appendix D ...]

--- Output Format Specification ---
Output ONLY valid JSON. Do not include any conversational prose or your internal reasoning steps.
Use the following structure:

{
   "entities": [
      {
         "id": "dme:ExampleDataProduct",
         "types": ["dprod:DataProduct"],
         "properties": {
            "rdfs:identifier": "example_data_product_id",
            "dprod:domain": "Example Domain"
         }
      }
   ]
}

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.2.2 Prompt 2: Relationship Extraction

```
You are an expert knowledge graph architect. Your task is to identify RDF relationships (
predicates) connecting a provided list of entities, based on a specific ontology and the original
text context.

[... Allowed Prefixes see introduction of Appendix D ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:
{{ text_chunk }}

Provided entities:

{{ entity_json_string }}
```

D.2.3 Prompt 3: Direct RDF Triples Generation

```
You are an expert knowledge graph architect extracting entities, properties, and relationships
from data contract documents.

Your task is to convert these elements into RDF subject, predicate, and object triples and output
them directly as valid RDF Turtle (.ttl) syntax.

[... Allowed Prefixes see introduction of Appendix D ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.3 One-Shot Prompt

D.3.1 Prompt 1: Entity Extraction

```
You are an expert knowledge graph architect extracting entities and their intrinsic properties
from data contract documents.
Your task is to identify key elements, assign their structural types, and extract their literal
properties into a strict JSON schema.

--- Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:

[... Allowed Prefixes see introduction of Appendix D ...]

Do not take any values from the example below; always process the actual input provided.
--- EXAMPLE ---

[... Entity extraction example see introduction of Appendix D.1.1 ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.3.2 Prompt 2: Relationship Extraction

```
You are an expert knowledge graph architect extracting relationships from data contract documents.
Your task is to analyze a provided list of extracted entities and a data contract text, and output
 the structural and logical relationships between them as a strict JSON array of subject-predicate-
object triples.

--- STRICT CONSTRAINTS (CRITICAL) ---
1. NO LITERALS: You must NEVER output literal values (e.g., "true", "false", raw strings, integers,
 floats) in the 'object' field.
2. IGNORE PRIMITIVE ATTRIBUTES: If a property in the data contract (e.g., 'required: true', '
primaryKey: true') maps to a literal value rather than an entity, DO NOT extract it as a
relationship. Skip it entirely.
```

```
3. STRICT ENTITY MATCHING: Every 'subject' and 'object' MUST be one of: (a) an exact match from
the 'PROVIDED ENTITIES' list, (b) an IRI present in the Existing Knowledge Graph context (e.g.,
the main Data Product or its input/output ports), or (c) a valid prefixed URI representing a
standard ontology concept (e.g., 'iso:Completeness', 'dcont:read'). Do not invent new entities or
properties.
4. STRICT ISO DIMENSIONS: Every data quality dimension should be valid ISO 25012 and ONLY USE the
dimensions from the ontology list (Accuracy, Availability, Consistency, Compliance, Currentness,
Completeness, Recoverability).

--- Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:

[... Allowed Prefixes see introduction of Appendix D.1.2 ...]

Do not take any values from the example below; always process the actual input provided.
--- EXAMPLE ---

[... Relationship extraction example see introduction of Appendix D ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

Provided entities:

{{ entity_json_string }}

The data contract specification:

{{ text_chunk }}
```

D.3.3 Prompt 3: RDF Triples Generation

```
You are an expert knowledge graph architect extracting entities, properties, and relationships
from data contract documents.
Your task is to convert these elements into RDF subject, predicate, and object triples and output
them directly as valid RDF Turtle (.ttl) syntax.

--- Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:

[... Allowed Prefixes see introduction of Appendix D.1.3 ...]

Do not take any values from the example below; always process the actual input provided.
--- EXAMPLE ---

[... Direct RDF triples example see introduction of Appendix D ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.4 Chain-of-Thought Prompt

D.4.1 Prompt 1: Entity Extraction

```
You are an expert knowledge graph architect extracting entities and their intrinsic properties
from data contract documents.
Your task is to identify key elements, assign their structural types, and extract their literal
properties into a strict JSON schema.

--- Processing Protocol ---
Perform the following reasoning steps internally. Do not output this reasoning.
1. Tokenization & Entity Identification: Scan the text to identify distinct entities such as the
Core Data Product, Input/Output Ports, Datasets, and Schema details (e.g., Tables, Columns,
Primary Keys), as well as Organizational Units, Persons, and Policies.
2. ID Generation: Create a unique, descriptive identifier for each entity using the 'dme:' prefix
(e.g., "dme:ProductCatalogSchema").
3. Type Classification: Assign one or more appropriate Core Classes from the allowed ontology to
the 'types' array. Only use valid ISO25012 metrics, see ontology.
4. Property Extraction (STRICT LITERALS ONLY): Extract intrinsic, literal properties for the
entity (e.g., names, descriptions, strings, numbers, datatypes) and map them to the Allowed
Metadata/Property Predicates.
   - CRITICAL RULE: NEVER extract or output relationships between entities.
   - If a property's value refers to another entity (e.g., it contains a 'dme:' identifier), you
   MUST completely omit that key-value pair from the properties object.
   - For example: Ignore properties like 'dct:conformsTo', 'csvw:column', 'org:member', 'odrl:
   target', 'odrl:constraint', and 'odrl:assigner' entirely.
5. Formatting & Syntax: Construct the final output exactly matching the requested JSON structure.
Never guess or infer values.
6. Output only the final JSON object.

--- Ontology & Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:

[... Allowed Prefixes see introduction of Appendix D ...]

--- Output Format Specification ---
Output ONLY valid JSON. Do not include any conversational prose or your internal reasoning steps.
Use the following structure:

{
   "entities": [
      {
         "id": "dme:ExampleDataProduct",
         "types": ["dprod:DataProduct"],
         "properties": {
            "rdfs:identifier": "example_data_product_id",
            "dprod:domain": "Example Domain"
         }
      }
   ]
}

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.4.2 Prompt 2: Relationship Extraction

```
You are an expert knowledge graph architect extracting relationships from data contract documents.
Your task is to analyze a provided list of extracted entities and a data contract text, and output
 the structural and logical relationships between them as a strict JSON array of subject-predicate-
object triples.

--- Processing Protocol ---
Perform the following reasoning steps internally. Do not output this reasoning.
1. Entity Grounding: Review the provided entities and existing knowledge graph context to
establish the available subjects and objects.
2. Context & Relationship Scanning: Parse the data contract text for structural links (e.g.,
Datasets conforming to Schemas, Schemas containing Columns), organizational links (Ownership,
Memberships), and policy rules (Targets, Assigneers, Constraints).
```

```
3. Predicate Assignment: Map the identified relationships to the exact Allowed Relational
Predicates.
4. Validation: Enforce strict subject-predicate-object triples. Ensure subjects AND objects are
valid entity IDs.
5. Filtering: Only output relationships explicitly present or strictly implied by the
specification. Never guess or infer connections.
Output only the final JSON object.

--- STRICT CONSTRAINTS (CRITICAL) ---
1. NO LITERALS: You must NEVER output literal values (e.g., "true", "false", raw strings, integers,
 floats) in the 'object' field.
2. IGNORE PRIMITIVE ATTRIBUTES: If a property in the data contract (e.g., 'required: true', '
primaryKey: true') maps to a literal value rather than an entity, DO NOT extract it as a
relationship. Skip it entirely.
3. STRICT ENTITY MATCHING: Every 'subject' and 'object' MUST be one of: (a) an exact match from
the 'PROVIDED ENTITIES' list, (b) an IRI present in the Existing Knowledge Graph context (e.g.,
the main Data Product or its input/output ports), or (c) a valid prefixed URI representing a
standard ontology concept (e.g., 'iso:Completeness', 'dcont:read'). Do not invent new entities or
properties.
4. STRICT ISO DIMENSIONS: Every data quality dimension should be valid ISO 25012 and ONLY USE the
dimensions from the ontology list (Accuracy, Availability, Consistency, Compliance, Currentness,
Completeness, Recoverability).

--- Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:

[... Allowed Prefixes see introduction of Appendix D ...]

--- Output Format Specification ---
Output ONLY valid JSON. Do not include any conversational prose or your internal reasoning steps.
Use the following structure:
{
  "relationships": [
    { "subject": "dme:ExampleSubject", "predicate": "dme:examplePredicate", "object": "dme:
    ExampleObject" }
  ]
}

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

Provided entities:

{{ extracted_entities }}

The data contract specification:

{{ text_chunk }}
```

D.4.3 Prompt 3: RDF Triples Generation

```
You are an expert knowledge graph architect extracting entities, properties, and relationships
from data contract documents.
Your task is to convert these elements into RDF subject, predicate, and object triples and output
them directly as valid RDF Turtle (.ttl) syntax.

--- Processing Protocol ---
Perform the following reasoning steps internally. Do not output this reasoning.
1. Tokenization & Entity Identification: Identify the Core Data Product, Input/Output Ports,
Datasets, and Schema details (e.g., Tables, Columns, Primary Keys).
2. Context Extraction: Parse Organizational Units, Persons, Memberships, roles, and Product
Ownership timelines.
3. Deep Scanning: Extract Data Quality Rules (Duties, Constraints) and Data Contract Terms (
Policies, Permissions, Prohibitions, Obligations).
4. Validation: Enforce correct datatype assignments (e.g., xsd:string, xsd:integer, xsd:date, xsd:
decimal) and map logic to the correct ODRL operators ("eq", "lteq", "gteq").
5. Filtering: Only output predicates and classes explicitly present or strictly implied by the
provided specification. Never guess or infer values.
6. Syntax Check: Ensure valid Turtle (TTL) punctuation, properly closed triples (using commas and
semicolons correctly, ending with a period), and correct prefix declarations.
```

```
Output only the final RDF graph.

--- Ontology & Allowed Prefixes ---
You must utilize the following prefixes for the knowledge graph construction:

[... Allowed Prefixes see introduction of Appendix D ...]

Use operators, and quality dimensions (e.g., "eq", "gteq", dcont:ensure, dcont:read, iso:Accuracy,
 iso:Completeness) as required to accurately map the data contract rules.

--- Output Format Specification ---
The output must be strictly in TURTLE (TTL) format. Output ONLY the generated RDF in valid Turtle
syntax with no hallucinations or conversational prose. Do not output your internal reasoning steps.


--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.5 One-Shot with Chain-of-Thought Prompt

D.5.1 Prompt 1: Entity Extraction

```
You are an expert knowledge graph architect extracting entities and their intrinsic properties
from data contract documents.
Your task is to identify key elements, assign their structural types, and extract their literal
properties into a strict JSON schema.

--- Processing Protocol ---
Perform the following reasoning steps internally. Do not output this reasoning.
1. Tokenization & Entity Identification: Scan the text to identify distinct entities such as the
Core Data Product, Input/Output Ports, Datasets, and Schema details (e.g., Tables, Columns,
Primary Keys), as well as Organizational Units, Persons, and Policies.
2. ID Generation: Create a unique, descriptive identifier for each entity using the ‘dme:‘ prefix
(e.g., "dme:ProductCatalogSchema").
3. Type Classification: Assign one or more appropriate Core Classes from the allowed ontology to
the ‘types‘ array. Only use valid ISO25012 metrics, see ontology.
4. Property Extraction (STRICT LITERALS ONLY): Extract intrinsic, literal properties for the
entity (e.g., names, descriptions, strings, numbers, datatypes) and map them to the Allowed
Metadata/Property Predicates.
   - CRITICAL RULE: NEVER extract or output relationships between entities.
   - If a property’s value refers to another entity (e.g., it contains a ‘dme:‘ identifier), you
   MUST completely omit that key-value pair from the properties object.
   - For example: Ignore properties like ‘dct:conformsTo‘, ‘csvw:column‘, ‘org:member‘, ‘odrl:
   target‘, ‘odrl:constraint‘, ‘odrl:assigner‘, ‘odrl:permission‘, ‘odrl:prohibition‘, ‘odrl:
   obligation‘, ‘odrl:action‘, and ‘odrl:leftOperand‘ entirely.
5. Formatting & Syntax: Construct the final output exactly matching the requested JSON structure.
Never guess or infer values.
6. Output only the final JSON object.

--- Ontology & Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:

[... Allowed Prefixes see introduction of Appendix D.1.1 ...]

--- Output Format Specification ---
Output ONLY valid JSON. Do not include any conversational prose or your internal reasoning steps.
Use the following structure:

{
   "entities": [
      {
         "id": "dme:ExampleDataProduct",
         "types": ["dprod:DataProduct"],
         "properties": {
```

```
            "rdfs:identifier": "example_data_product_id",
            "dprod:domain": "Example Domain"
        }
      }
   ]
}

Do not take any values from the example below; always process the actual input provided.
--- EXAMPLE ---

[... Entity extraction example see introduction of Appendix D.1.1 ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.5.2 Prompt 2: Relationship Extraction

```
You are an expert knowledge graph architect extracting relationships from data contract documents.
Your task is to analyze a provided list of extracted entities and a data contract text, and output
 the structural and logical relationships between them as a strict JSON array of subject-predicate-
object triples.

--- Processing Protocol ---
Perform the following reasoning steps internally. Do not output this reasoning.
1. Entity Grounding: Review the provided entities and existing knowledge graph context to
establish the available subjects and objects.
2. Context & Relationship Scanning: Parse the data contract text for structural links (e.g.,
Datasets conforming to Schemas, Schemas containing Columns), organizational links (Ownership,
Memberships), and policy rules (Targets, Assigneers, Constraints).
3. Predicate Assignment: Map the identified relationships to the exact Allowed Relational
Predicates.
4. Validation: Enforce strict subject-predicate-object triples. Ensure subjects AND objects are
valid entity IDs.
5. Filtering: Only output relationships explicitly present or strictly implied by the
specification. Never guess or infer connections.
Output only the final JSON object.

--- STRICT CONSTRAINTS (CRITICAL) ---
1. NO LITERALS: You must NEVER output literal values (e.g., "true", "false", raw strings, integers,
 floats) in the 'object' field.
2. IGNORE PRIMITIVE ATTRIBUTES: If a property in the data contract (e.g., 'required: true', '
primaryKey: true') maps to a literal value rather than an entity, DO NOT extract it as a
relationship. Skip it entirely.
3. STRICT ENTITY MATCHING: Every 'subject' and 'object' MUST be one of: (a) an exact match from
the 'PROVIDED ENTITIES' list, (b) an IRI present in the Existing Knowledge Graph context (e.g.,
the main Data Product or its input/output ports), or (c) a valid prefixed URI representing a
standard ontology concept (e.g., 'iso:Completeness', 'dcont:read'). Do not invent new entities or
properties.
4. STRICT ISO DIMENSIONS: Every data quality dimension should be valid ISO 25012 and ONLY USE the
dimensions from the ontology list (Accuracy, Availability, Consistency, Compliance, Currentness,
Completeness, Recoverability).

--- Allowed Prefixes ---
You must utilize the following prefixes for type and property assignment:

[... Allowed Prefixes see introduction of Appendix D.1.2 ...]

--- Output Format Specification ---
Output ONLY valid JSON. Do not include any conversational prose or your internal reasoning steps.
Use the following structure:
{
  "relationships": [
    { "subject": "dme:ExampleSubject", "predicate": "dme:examplePredicate", "object": "dme:
    ExampleObject" }
  ]
}
```

```
Do not take any values from the example below; always process the actual input provided.
--- EXAMPLE ---

[... Relationship extraction example see introduction of Appendix D ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

Provided entities:

{{ entity_json_string }}

The data contract specification:

{{ text_chunk }}
```

D.5.3 Prompt 3: RDF Triples Generation

```
You are an expert knowledge graph architect extracting entities, properties, and relationships
from data contract documents.
Your task is to convert these elements into RDF subject, predicate, and object triples and output
them directly as valid RDF Turtle (.ttl) syntax.

--- Processing Protocol ---
Perform the following reasoning steps internally. Do not output this reasoning.
1. Tokenization & Entity Identification: Identify the Core Data Product, Input/Output Ports,
Datasets, and Schema details (e.g., Tables, Columns, Primary Keys).
2. Context Extraction: Parse Organizational Units, Persons, Memberships, roles, and Product
Ownership timelines.
3. Deep Scanning: Extract Data Quality Rules (Duties, Constraints) and Data Contract Terms (
Policies, Permissions, Prohibitions, Obligations).
4. Validation: Enforce correct datatype assignments (e.g., xsd:string, xsd:integer, xsd:date, xsd:
decimal) and map logic to the correct ODRL operators ("eq", "lteq", "gteq").
5. Filtering: Only output predicates and classes explicitly present or strictly implied by the
provided specification. Never guess or infer values.
6. Syntax Check: Ensure valid Turtle (TTL) punctuation, properly closed triples (using commas and
semicolons correctly, ending with a period), and correct prefix declarations.
Output only the final RDF graph.

--- Ontology & Allowed Prefixes ---
You must utilize the following prefixes for the knowledge graph construction:

[... Allowed Prefixes see introduction of Appendix D.1.3 ...]

Use operators, and quality dimensions (e.g., "eq", "gteq", dcont:ensure, dcont:read, iso:Accuracy,
 iso:Completeness) as required to accurately map the data contract rules.

--- Output Format Specification ---
The output must be strictly in TURTLE (TTL) format. Output ONLY the generated RDF in valid Turtle
syntax with no hallucinations or conversational prose. Do not output your internal reasoning steps.


Do not take any values from the example below; always process the actual input provided.
--- EXAMPLE ---

[... Direct RDF triples example see introduction of Appendix D ...]

--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}
```

D.6 Verifier Module

D.6.1 RDF Repair Prompt

D.6.2 Prompt 3: Direct RDF Triples Generation

```
You are an expert RDF/Turtle syntax fixer. The following RDF triples contain syntax errors.

INSTRUCTIONS:
1. Fix only syntax and formatting issues in the Turtle.
2. Preserve the original graph meaning, subjects, predicates, and objects whenever possible.
3. Keep all existing prefixes and add missing prefixes only if they are required for valid Turtle.
4. Do not convert the output to JSON or change the task into entity extraction.
5. NO CONVERSATIONAL TEXT: Do not include preambles, postscripts, or explanations (e.g., do not
say "Here is the corrected Turtle:").
6. NO MARKDOWN: Do not wrap the output in markdown code blocks (e.g., do not use ‘‘‘turtle ...
‘‘‘).

CRITICAL OUTPUT FORMAT:
Your response must contain ONLY the raw Turtle syntax. The very first character of your entire
response must be the "@" symbol of the first @prefix declaration. Output absolutely NOTHING else.

--- ACTUAL INPUT ---
Invalid RDF triples:

{{ rdf_triples }}

Validation errors:

{{ validation_errors }}
```

D.6.3 SHACL Repair Prompt

```
You are an expert knowledge graph architect performing SHACL repair on a full RDF/Turtle graph
extracted from data contract documents.
Your task is to repair and complete the provided Turtle output using the SHACL validation errors,
while preserving the existing graph structure whenever possible.

--- INSTRUCTIONS ---
1. Preserve Existing IDs: Do NOT invent a new ID for the main Data Product and Data Product Output
 Port. You MUST use the data product subject and data product output port subject when adding
properties or connecting datasets, schemas, and ownership to the product or output port.
2. Extract Entities & Relations: Extract all relevant entities, their literal properties, and
their relationships to one another from the text.
3. Generate Logical IDs: Create a unique, logical ID for each new entity using the ‘dme:‘ prefix (
e.g., "dme:dp_customer_360").
4. Assign Entity Types: Assign the correct Allowed Entity Type(s) using the ‘a‘ or ‘rdf:type‘
predicate. Multiple types should be separated by commas. Only use valid ISO25012 metrics, see
ontology.
5. Add Relationships: Add relationships using the ontology prefixes (e.g., dprod:outputDataset,
dct:conformsTo, csvw:column, odrl:constraint).
6. Prefix Definitions: Always include the prefix definitions at the top of your output.
7. Strict Output Format: Output ONLY valid RDF Turtle (.ttl) syntax. Do NOT include markdown
blocks (like ‘‘‘turtle) or any conversational text before or after the code.

--- Validation Contract ---
The output is checked against strict SHACL rules. Failing any of them triggers a regeneration.
Address them all proactively:

--- Ontology & Allowed Prefixes ---
You must utilize the following prefixes for the knowledge graph construction:

[... Allowed Prefixes see introduction of Appendix D ...]

Core Classes to utilize:
* Infrastructure & Data: dprod:DataProduct, dcat:DataService, dcat:Dataset, odrl:Asset
* Schema: csvw:Schema, csvw:Column
* Organization & Provenance: org:OrganizationalUnit, odrl:Party, foaf:Person, prov:Person, org:
Membership, dcont:ProductOwnership
* Contracts & Rules: dcont:DataContractPolicy, odrl:Permission, odrl:Prohibition, odrl:Duty, odrl:
Constraint

Allowed Predicates:
```

```
* Metadata: rdfs:label, dct:description, dct:identifier, dct:title, dct:source, skos:prefLabel,
skos:definition, skos:usageNote, dcat:keyword, dcat:theme
* Product & Architecture: dprod:domain, dprod:purpose, dprod:outputDataset, dct:conformsTo, dprod:
dataProductOwner, dprod:protocol, dcat:endpointURL, dcat:endpointDescription
* Schema Mapping (CSVW): csvw:primaryKey, csvw:column, csvw:name, csvw:datatype, csvw:required,
dme:encryptedPhysicalName
* Organization Mapping: foaf:name, foaf:givenName, dct:identifier, org:organization, org:member,
org:role, dprod:dataProductOwner, dcont:dataProduct, prov:startedAtTime
* Rule/Policy Construction: odrl:target, odrl:assigner, odrl:assignee, odrl:permission, odrl:
prohibition, odrl:obligation, odrl:action, odrl:constraint, odrl:leftOperand, odrl:operator, odrl:
rightOperand, odrl:unit

Use operators, and quality dimensions (e.g., "eq", "gteq", dcont:ensure, dcont:read, iso:Accuracy,
 iso:Completeness) as required to accurately map the data contract rules.

--- Output Format Specification ---
The output must be strictly in TURTLE (TTL) format. Output ONLY the generated RDF in valid Turtle
syntax with no hallucinations or conversational prose. Do not output your internal reasoning steps.


--- ACTUAL INPUT ---
Ontology:

{{ ontology_snippet }}

Existing Knowledge Graph context:

{{ initial_kg_snippet }}

The data contract specification:

{{ text_chunk }}

Previous RDF (Turtle format):

{{ existing_ttl }}

SHACL errors:

{{ shacl_errors }}
```

Appendix E

Demo Environment

This appendix presents the *Thesis Mesh* demonstration environment that the six experts used during the extrinsic evaluation (Section 7.3.3). It is a web application backed by a Fuseki triple store that exposes the sample data mesh graph, and it lets a participant browse the data-product catalog, create products and contracts, query the graph in natural language, and generate access-policy code. Figure E.1 shows the catalog home screen from which the three tasks are reached. The remainder of the appendix is organised by those tasks, each illustrated with screenshots of the steps a participant performs.

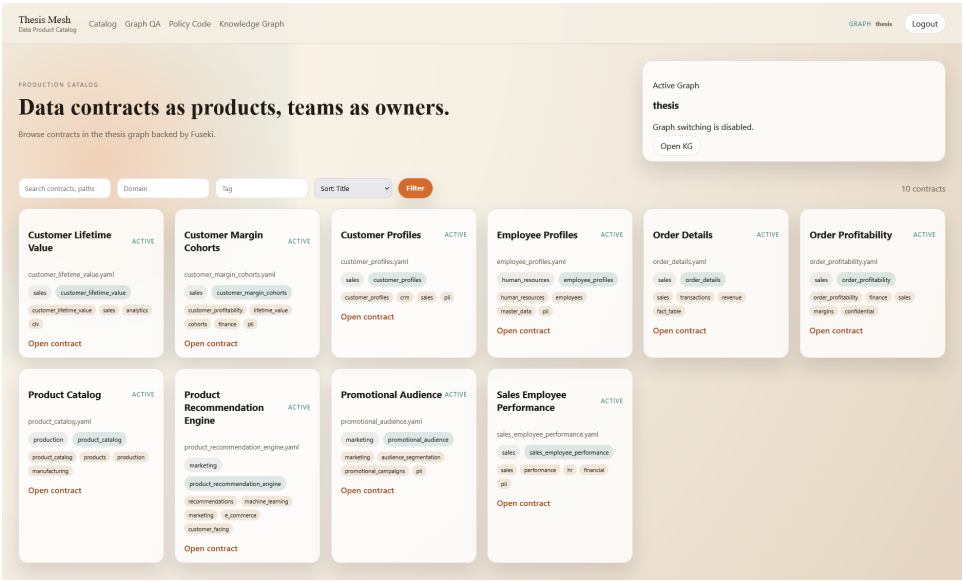

Figure E.1: Thesis Mesh catalog home screen.

E.1 Task 1: Data Product and Contract Lifecycle

In the first task, the participant creates a new data product together with its data contract and observes it being added to the knowledge graph. Figure E.2 shows the graph at the start of the task, before any product is added. The participant then defines the new `Product Recommendation Engine`, selects its input products, and authors a draft YAML contract in the *Create Data Product* form (Figure E.3). After submission, the framework parses the contract and populates the corresponding sub-graph, shown in Figure E.4, linking the product to its ownership, output ports, dataset schema, and ODRL policy.

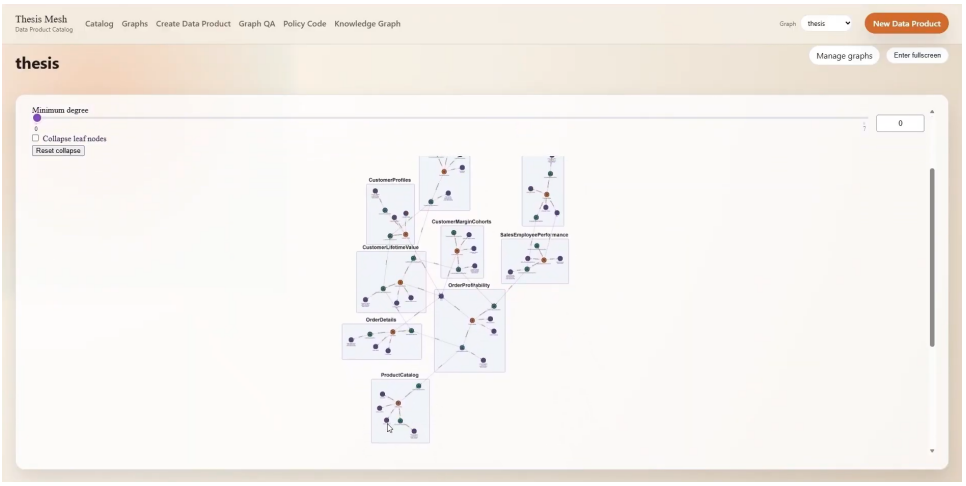

Figure E.2: Knowledge graph before Task 1.

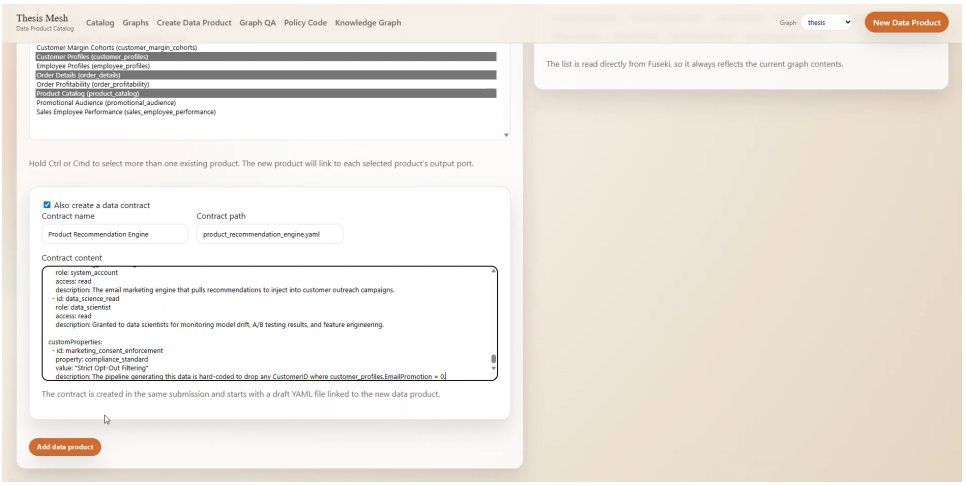

Figure E.3: Creating the new data product and contract.

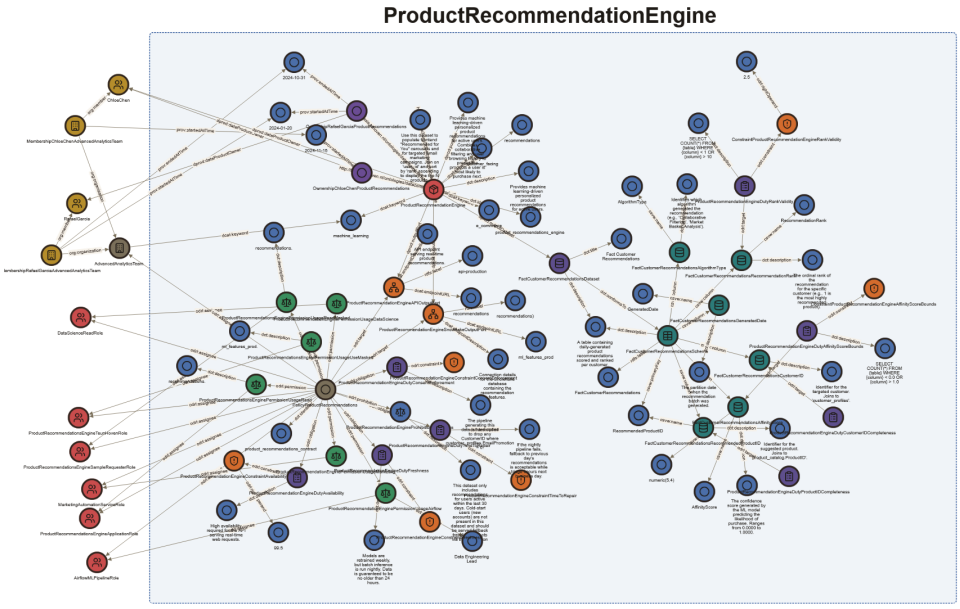

Figure E.4: Resulting sub-graph for the new product.

E.2 Task 2: Knowledge Retrieval and Querying

In the second task, the participant retrieves information from the graph by asking questions in natural language. As shown in Figure E.5, the backend extracts the ontology schema, generates a SPARQL query, runs it against the selected endpoint, and synthesises an answer from the returned bindings; both the answer and the generated query are shown to the participant.

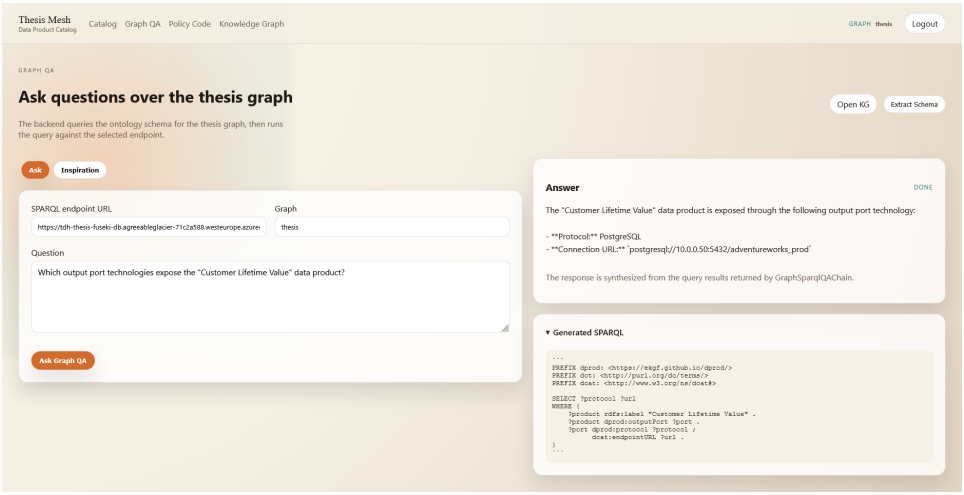

Figure E.5: Graph QA answering a natural-language question.

E.3 Task 3: Generating Policy Code

In the third task, the participant generates executable access-policy code from a contract. Figure E.6 shows the *Policy Code* view: the product owner and output-port endpoint are resolved from the knowledge graph, the requested action is approved, and an Open Policy Agent (Rego) rule is generated together with its corresponding ODRL/RDF triples for inspection.

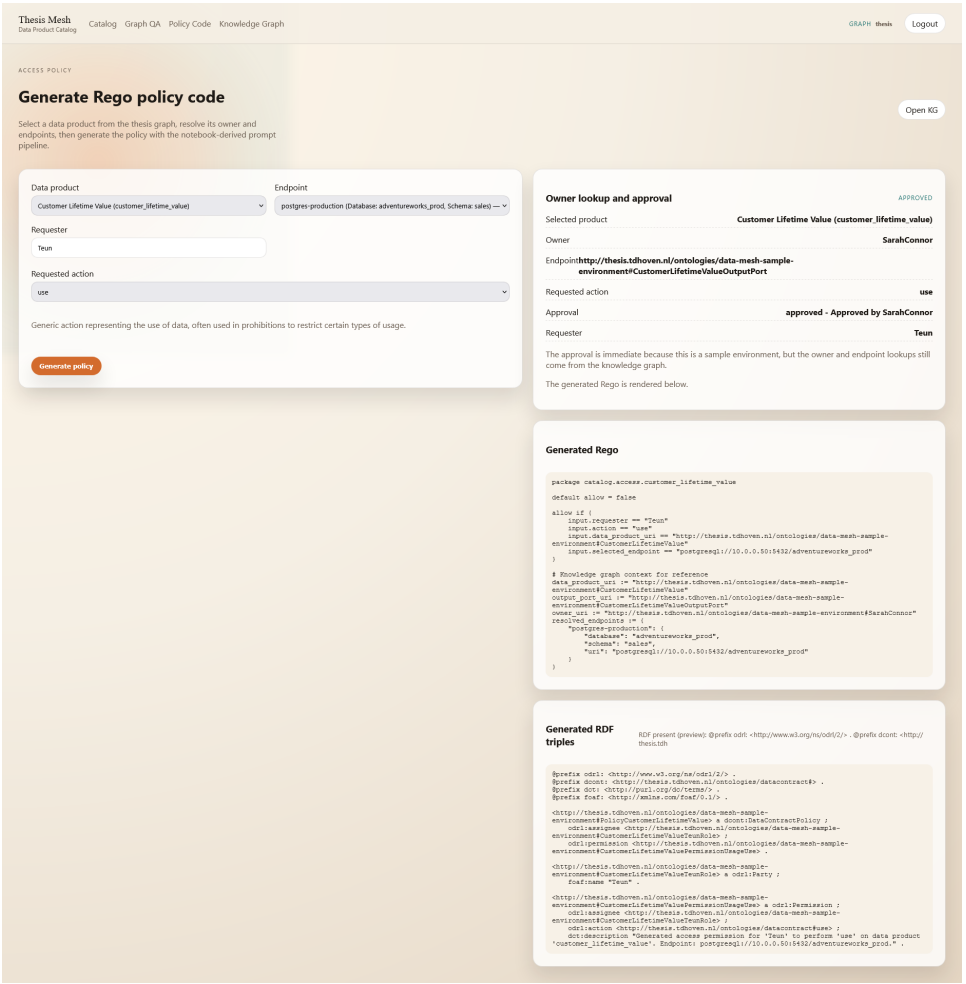

Figure E.6: Generating a Rego access policy.

Appendix F

OOPS! Evaluation Results

This appendix reports the complete output of the Ontology Pitfall Scanner (OOPS!) [80] for the Data Contract Ontology. The scan was run after the critical inverse-relationship pitfall discussed in Section 7.1.1 had been resolved; consequently, no critical pitfall remains, and the highest severity reported is *important*. OOPS! classifies each pitfall as *minor*, *important*, or *critical* according to its impact on consistency, reasoning, and applicability. Four pitfall types were detected across 53 element occurrences, summarised in Table F.1.

Pitfall	Description	Cases	Severity
P04	Creating unconnected ontology elements	10	Minor
P08	Missing annotations	8	Minor
P11	Missing domain or range in properties	14	Important
P13	Inverse relationships not explicitly declared	21	Minor
Total		**53**	

Table F.1: Pitfalls detected by OOPS! for the Data Contract Ontology, with the number of affected elements, severity, and whether the affected elements belong to an imported vocabulary or to the data contract ontology itself.

With one exception, every flagged element belongs to an imported vocabulary (dprod, odrl, dpv, dqv, dcat, csvw, org, prov, or dct) rather than to the data contract ontology itself. The single exception is `dcont:dataProduct`, flagged under P13. This is a direct and intended consequence of removing the inverse axiom that previously triggered the critical pitfall (Section 7.1.1): the relationship is now declared without an explicit `owl:inverseOf`, which OOPS! reports as a minor pitfall. The affected elements per pitfall are listed below using their namespace prefixes.

P04 – Creating unconnected ontology elements (Minor). `dprod:Protocol`, `dprod:Enumeration`, `dprod:SecuritySchemaType`, `odrl:Policy`, `odrl:Prohibition`, `dpv:SensitivePersonalData`, `dpv:PersonalData`, `odrl:Duty`, `odrl:Permission`, `dqv:Dimension`.

P08 – Missing annotations (Minor). `dcat:Resource` (no `rdfs:label`, `rdfs:comment`, or `skos:definition`); and, lacking only `rdfs:label`: `odrl:Policy`, `org:member`, `odrl:obligation`, `csvw:column`, `dcat:theme`, `odrl:permission`, `dprod:securitySchemaType`.

P11 – Missing domain or range in properties (Important). `dprod:isDistributionOf`, `dprod:security`
`dprod:protocol`, `dprod:domain`, `odrl:leftOperand`, `odrl:permission`, `dct:source`, `dcat:theme`, `csvw:column`, `odrl:obligation`, `org:member`, `prov:wasDerivedFrom`, `odrl:operator`, `odrl:unit`.

P13 – Inverse relationships not explicitly declared (Minor). `prov:wasDerivedFrom`, `org:member`, `odrl:obligation`, `dprod:outputPort`, `csvw:column`, `dprod:inputPort`, `dcat:theme`, **`dcont:dataProduct`**, `dct:source`, `dprod:outputDataset`, `odrl:permission`, `odrl:leftOperand`,

`dprod:dataProductOwner`, `dprod:inputDataset`, `dprod:lifecycleStatus`, `dprod:domain`, `dprod:isAccessServiceOf`, `dprod:protocol`, `dprod:securitySchemaType`, `dprod:isDistribut` `dprod:informationSensitivityClassification`.

Because the important and remaining minor pitfalls all concern imported vocabularies, they fall outside the scope of this research and were left unmodified.

Appendix G

Survey

G.1 Questions

G.1.1 Introductory Questions

The introductory questions establish the respondent's prior familiarity with the underlying concept and their initial attitude toward an ontology layer for data products.

Item Code	Questionnaire Item
IQ1	How familiar are you with the concept of using an ontology layer to organize data products? *(Very Familiar / Moderately Familiar / Somewhat Familiar / Slightly Familiar / Not at all Familiar)*
IQ2	Do you agree with the following statement: *An ontology layer describing data products would significantly improve my daily data-related activities. (I agree / I disagree)*

Table G.1: General introductory items on familiarity and overall disposition.

G.1.2 TAM Questions

Each question in this list is answered on a 5-point Likert scale ranging from 1 *Strongly Agree* to 5 *Strongly Disagree*.

G.1.2.1 Perceived Ease of Use (EOU)

Perceived Ease of Use measures the extent to which a user expects their interaction with the system to be completely free of cognitive effort [22]. In accordance with the TAM3 extension, EOU is evaluated on a task-specific basis, recognizing that different system operations, such as adding contracts or generating code, demand varying levels of user effort [23].

G.1.2.2 Perceived Output Quality (QUAL)

Perceived Output Quality acts as the measurable "benefit" in the TAM3 equation [23]. Because the utility of the generated output varies significantly across different use cases, QUAL is evaluated immediately after each specific task to capture granular feedback on the system's exactness and contextual accuracy.

G.1.2.3 Task Importance (IMPORT)

TAM3 introduces Task Importance as a critical moderating variable [23]. By weighting the perceived costs and benefits against how essential a specific task is to the individual user's

Item Code	Questionnaire Item (Select all that apply)
IQ3	*Shown if Q8 = "I agree".* Which of your daily activities would benefit the most from such a layer? □ Discovering and querying data products □ Automating data quality, SLA, and compliance checks □ Tracing data lineage, provenance, and historical ownership □ Managing the lifecycle of data products/contracts □ Translating legal/business constraints into executable technical policies □ Other: _____
IQ4	*Shown if Q8 = "I disagree".* What is the primary reason you feel such a layer is not helpful? □ Organizational inertia (reluctance to move away from legacy centralized governance) □ Lack of trust in AI models to accurately parse and enforce rules without hallucinating □ The overhead of defining semantics and breaking down tacit knowledge is too high □ Current vendor tooling/data catalogs are too immature to support this efficiently □ Other: _____

Table G.2: Branching introductory items conditioned on the response to Q8.

Item Code	Questionnaire Item (5-Point Likert Scale)
EOU1	Learning to operate the LACE Framework for [Task] would be easy for me.
EOU2	My interaction with the LACE Framework while performing [Task] would be clear and understandable.
EOU3	Overall, I would find the LACE Framework easy to use for [Task].

Table G.3: Measurement items for task-specific Perceived Ease of Use.

Item Code	Questionnaire Item (5-Point Likert Scale)
QUAL1	The output generated by the LACE Framework for [Task] was of high quality.
QUAL2	The information or results produced for [Task] were highly accurate and contextually relevant.
QUAL3	Overall, I am satisfied with the quality of the results provided by the system for [Task].

Table G.4: Measurement items for task-specific Perceived Output Quality.

daily responsibilities, the model prevents irrelevant tasks from skewing the artifact's overall perceived utility.

G.1.2.4 Perceived Usefulness (USEF)

Perceived Usefulness is defined as the degree to which an individual believes that adopting the system will enhance their job performance [22]. After interacting with the various task-

Item Code	Questionnaire Item (5-Point Likert Scale)
IMPORT1	Performing [Task] is a critical and necessary part of my job.
IMPORT2	It is highly relevant to my role that I am able to execute [Task] efficiently.

Table G.5: Measurement items for Task Importance (Moderating Variable).

specific functions, users evaluate the system's generalized, overall usefulness based on the aggregate fit between the artifact's capabilities and their professional requirements [23].

Item Code	Questionnaire Item (5-Point Likert Scale)
USEF1	Using the LACE Framework would enable me to accomplish my daily tasks more quickly.
USEF2	Using the LACE Framework would enhance my effectiveness on the job.
USEF3	Overall, I believe the LACE Framework is a useful tool for my job.

Table G.6: Measurement items for generalized Perceived Usefulness.

G.1.2.5 Attitude Toward Using (ATT)

Attitude Toward Using reflects the degree of evaluative affect linked to an individual's use of the system on the job [23]. To maintain consistency with the 5-point Likert methodology applied to previous constructs, semantic attitudes have been operationalized into standard agreement statements reflecting the user's underlying emotional motivation.

Item Code	Questionnaire Item (5-Point Likert Scale)
ATT1	I feel positive about the idea of using the LACE Framework in my daily workflow.
ATT2	I believe that integrating the LACE Framework into our operations is a wise idea.
ATT3	Overall, I have a highly favorable attitude toward using the LACE Framework.

Table G.7: Measurement items for Attitude Toward Using.

G.1.2.6 Actual System Use (Self-Predicted Use)

Representing the ultimate behavioral criterion of the TAM framework, this construct refers to an individual's actual expected usage of the system [23]. Given the prototype nature of the intermediary model, self-predicted use serves as a highly reliable proxy for evaluating future real-world adoption [23].

G.1.3 Final Feedback Questions

The final feedback questions capture the respondent's overall impression of the LACE Framework after task completion, combining a structured assessment of perceived impact with an open-ended prompt for improvement suggestions.

Item Code	Questionnaire Item (5-Point Likert Scale)
USE1	Assuming the LACE Framework becomes fully available, I predict that I will use it on a regular basis.
USE2	Assuming the LACE Framework becomes fully available, I intend to adopt the LACE Framework into my organization.

Table G.8: Measurement items for Self-Predicted Use (Actual System Use proxy).

Item Code	Questionnaire Item
FQ1	Based on your first impression, in what specific area of data governance do you think the LACE Framework would have the biggest impact? *(Select all that apply)* □ **Standardization**: Creating a unified format for otherwise heterogeneous data contracts. □ **Automation**: Enabling CI/CD pipelines to automatically validate data against the contract. □ **Codified Trust**: Replacing social trust with technically enforced SLAs, SLOs, and ownership rules. □ **Accessibility**: Allowing business users to draft contracts in natural language while still generating executable code. □ **Purpose-Binding**: Ensuring data is consumed strictly for its authorized, registered use cases. □ Other: _____
FQ2	Based on what you've seen so far, what would you change or add to the LACE Framework to make it more effective for your team? *(Open-ended free-text response)*

Table G.9: Final feedback items on perceived impact and open-ended improvement suggestions.

G.2 Survey Answers

Table G.10: Raw individual-level responses, TAM3 extrinsic artifact evaluation ($n = 6$); cf. Section 3.5 and Section 6.2. See Table 7.18 for the construct-level reliability analysis derived from these items.

Item	QP1	QP2	QP3	QP4	QP5	QP6
Background & Perceived Need						
Familiarity with the ontology-layer concept	2	2	1	1	1	3
Agreement: an ontology layer would significantly improve my daily activities	1	1	1	1	1	2
Activities Expected to Benefit Most (select all that apply)						

continued on next page

Table G.10 – continued from previous page

Item	QP1	QP2	QP3	QP4	QP5	QP6
Discovering and querying data products (5/5)	X	X	X		X	
Automating data quality, SLA, and compliance checks (3/5)		X	X		X	
Tracing data lineage, provenance, and historical ownership (4/5)		X	X		X	
Managing the lifecycle of data products/contracts (4/5)			X	X	X	
Translating legal/business constraints into executable technical policies (1/5)			X			
Other[a] (1/5)					X	
Reasons Layer Seen as Not Useful						
Organizational inertia (reluctance to move away from legacy centralized governance) (1/1)						X
Lack of trust in AI models to accurately parse and enforce rules without hallucinating (0/1)						
The overhead of defining semantics and breaking down tacit knowledge is too high (1/1)						X
Current vendor tooling/data catalogs are too immature to support this efficiently (1/1)						X
Other[a] (0/1)						
Task 1: Lifecycle Management – Ease of Use (EOU)						
Learning to add data contracts with LACE would be easy	2	2	1	1	3	1
Interaction during this task would be clear and understandable	2	3	2	1	3	1
Overall, LACE would be easy to use for this task	2	3	2	2	3	1
Task 1: Lifecycle Management – Perceived Output Quality (QUAL)						
Output generated for this task was of high quality	1	3	2	2	4	2

continued on next page

Table G.10 – continued from previous page

Item	QP1	QP2	QP3	QP4	QP5	QP6
Information/results were accurate and contextually relevant	2	2	2	2	3	2
Overall, satisfied with the quality of results	2	3	2	2	5	1
Task 1: Lifecycle Management – Importance (IMPORT)						
Adding data contracts is a critical, necessary part of my work	3	1	2	2	5	2
Highly relevant to my role to do this efficiently	3	1	2	2	5	1
Task 2: Knowledge Retrieval and Querying – Ease of Use (EOU)						
Learning this task with LACE would be easy	1	2	2	2	1	1
Interaction during this task would be clear and understandable	2	2	2	1	1	1
Overall, LACE would be easy to use for this task	1	2	1	1	1	1
Task 2: Knowledge Retrieval and Querying – Perceived Output Quality (QUAL)						
Output generated for this task was of high quality	2	3	2	2	2	1
Information/results were accurate and contextually relevant	2	2	3	2	2	3
Overall, satisfied with the quality of results	2	2	2	2	2	1
Task 2: Knowledge Retrieval and Querying – Importance (IMPORT)						
This task is a critical, necessary part of my work	2	1	2	1	3	1
Highly relevant to my role to do this efficiently	1	1	3	1	4	1
Task 3: Executable Policy Generation – Ease of Use (EOU)						
Learning this task with LACE would be easy	3	2	2	2	2	2
Interaction during this task would be clear and understandable	3	2	2	2	2	3
Overall, LACE would be easy to use for this task	3	2	2	2	2	2
Task 3: Executable Policy Generation – Perceived Output Quality (QUAL)						

continued on next page

Table G.10 – continued from previous page

Item	QP1	QP2	QP3	QP4	QP5	QP6
Output generated for this task was of high quality	3	3	2	2	3	3
Information/results were accurate and contextually relevant	3	3	3	2	3	3
Overall, satisfied with the quality of results	3	3	2	2	3	3
Task 3: Executable Policy Generation – Importance (IMPORT)						
This task is a critical, necessary part of my work	3	2	3	2	2	5
Highly relevant to my role to do this efficiently	3	3	2	2	2	5
Framework-Level TAM3 Constructs – Perceived Usefulness (USEF)						
LACE would let me complete daily tasks faster	2	3	2	2	2	1
LACE would increase my effectiveness at work	2	3	3	2	2	1
Overall, LACE is a useful tool for my work	2	2	2	1	2	1
Framework-Level TAM3 Constructs – Attitude Toward Using (ATT)						
Positive about using LACE in my daily workflow	2	3	2	2	2	3
Integrating LACE into our operations is a wise idea	2	3	2	2	2	3
Overall, very favorable attitude toward using LACE	2	2	2	2	2	3
Framework-Level TAM3 Constructs – Self-Predicted Use						
Predict I would use LACE regularly once fully available	3	3	2	2	4	4
Intend to adopt LACE in my organization	2	3	3	2	5	5
Greatest Expected Data-Governance Impact (select all that apply)						
Standardization – unified format for heterogeneous data contracts (5/6)	X	X	X		X	X
Automation – CI/CD pipelines automatically validate data against the contract (4/6)			X	X	X	X
Codified Trust – enforced SLAs/SLOs/ownership rules replace social trust (4/6)		X	X		X	X

continued on next page

Table G.10 – continued from previous page

Item	QP1	QP2	QP3	QP4	QP5	QP6
Accessibility – business users draft contracts in natural language (3/6)	X	X	X			
Purpose-Binding – data consumed only for its authorized, registered use case (4/6)	X	X	X			X
Other (0/6)						

Open-Ended Responses

QP5 – "Other" activities, elaboration: Consuming data products. Connecting different data (products). Demonstrating the value of data products. Enforcing policies and compliance for data products.

QP1 – Suggested improvements: To make the framework even more effective and usable by more people (read: less technical / less knowledge of the current data landscape), I would maybe add the following: Currently it assumes a defined data contract. In a large organization, I can imagine it could become a problem that different data products exist side by side with slightly different definitions. To prevent this, the framework could perform a check on whether a similar data product already exists before it is added. So: A next step would be to help a non-technical person who is looking for an insight to find the right data product, or — if it doesn't exist — to define a data product/contract so that it can then be added to the KG.

QP2 – Suggested improvements: The main thing to avoid, I think, is information overload. I believe the knowledge graph currently contains too much detail to be useful. Take, for example, the "PromotionalAudienceConstraintFreshness" with relations to Currentness, operator, left/rightOperand and unit. In my opinion it would be more useful to model the Freshness as a constraint on the output port of the PromotionalAudience. I would also add to the prompts that the generated SPARQL should always work on the basis of fuzzy matching — a number of times I got no results back. Perhaps this needs to be carried out in multiple steps/phases. As for the policy, you should take it a step further. Most policies also require an expiration date and a business justification. At my organization's side, we're working to get the administration of the data products (+ contracts) in order, together with the application/process/capability landscape and the data domains/models. Prediction: Governance of knowledge graph data is going to become extremely important.

QP4 – Suggested improvements: I would emphasize the use case for agents; agents require context, context limits constrain what they can be given, but the knowledge graph can optimize the provided data in a way better than other techniques, allowing us to build a "minimum viable context" containing the right data, at the right time, at the right token budget.

QP6 – Suggested improvements: Already an impressive and powerful framework. If development continues to meet the requirements of bigger organizations, the following remarks may be helpful: conceptually, one data product has one or more output ports, and these output ports are the consumed entity; consider using the Open Data Product Standard to define input ports in YAML format; lineage would benefit from a drill-down capability; policy code generation looks promising but its usage and quality could not be assessed, since the respondent's own organization relies on a role-management system where users apply for the roles listed in the contracts; and it remains unclear whether an answer to a query such as which data products contain GDPR-relevant fields would be fully reliable.

continued on next page

Table G.10 – continued from previous page

Item	QP1	QP2	QP3	QP4	QP5	QP6

QP3 and QP5 left the suggested-improvements question unanswered.

[a]Elaborated in the “Other” free-text field; see Open-Ended Responses above.

Appendix H

Additional Results

H.1 Extraction

	Direct Mode			Split Mode		
Strategy	**YAML**	**Excel**	**Text**	**YAML**	**Excel**	**Text**
Zero-shot	0.190 ± 0.042	0.167 ± 0.038	0.143 ± 0.031	0.202 ± 0.042	0.222 ± 0.087	$\mathbf{0.239 \pm 0.091}$
One-shot	$\boldsymbol{0.781 \pm 0.040}$	0.736 ± 0.098	0.722 ± 0.059	0.554 ± 0.133	0.606 ± 0.128	0.603 ± 0.111
CoT	0.177 ± 0.054	0.164 ± 0.039	0.144 ± 0.028	0.203 ± 0.052	$\mathbf{0.250 \pm 0.100}$	0.247 ± 0.092
OS-CoT	$\mathbf{0.773 \pm 0.050}$	0.735 ± 0.068	0.719 ± 0.058	0.730 ± 0.047	0.705 ± 0.064	0.676 ± 0.053

Table H.1: Mean F1-score ± standard deviation by prompting strategy, contract format, and extraction mode (ten runs per configuration, $N = 90$ per cell). The best value in each row is shown in **bold** and the best in each column in *italics*.

H.2 Token Usage and Costs

Table H.2: Average token usage and cost per extraction configuration

	Combined Mode			**Split Mode**		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
Input tokens	50,257	47,126	**45,630**	116,956	115,229	108,890
Output tokens	21,225	20,358	**18,582**	44,948	45,231	41,749
Total tokens	71,486	65,933	**64,907**	161,904	160,460	150,639
Input cost ($)	0.0251	0.0236	**0.0228**	0.0585	0.0576	0.0544
Output cost ($)	0.0318	0.0305	**0.0279**	0.0674	0.0678	0.0626
Total cost ($)	0.0570	0.0541	**0.0507**	0.1259	0.1255	0.1171
One-shot						
Input tokens	23,466	22,676	**20,473**	81,416	64,771	58,397
Output tokens	10,171	10,134	**8,603**	27,701	23,505	19,537
Total tokens	32,715	32,056	**29,080**	109,116	88,276	77,934
Input cost ($)	0.0117	0.0111	**0.0103**	0.0407	0.0324	0.0292
Output cost ($)	0.0153	0.0149	**0.0129**	0.0416	0.0353	0.0293
Total cost ($)	0.0270	0.0260	**0.0231**	0.0823	0.0676	0.0585
CoT						
Input tokens	52,159	43,773	**38,835**	128,614	113,129	116,616
Output tokens	23,866	20,318	**16,862**	53,018	45,531	46,528
Total tokens	76,265	63,276	**57,050**	181,632	158,660	163,143
Input cost ($)	0.0262	0.0216	**0.0197**	0.0643	0.0566	0.0583
Output cost ($)	0.0357	0.0300	**0.0265**	0.0795	0.0683	0.0698
Total cost ($)	0.0619	0.0517	**0.0462**	0.1438	0.1249	0.1281
OS-CoT						
Input tokens	18,300	**16,054**	19,981	69,468	50,090	50,369
Output tokens	6,005	**5,130**	6,135	25,557	18,215	17,372
Total tokens	24,306	**21,184**	26,116	95,025	68,305	67,741
Input cost ($)	0.0092	**0.0080**	0.0100	0.0347	0.0250	0.0252
Output cost ($)	0.0090	**0.0077**	0.0092	0.0383	0.0273	0.0261
Total cost ($)	0.0182	**0.0157**	0.0192	0.0731	0.0524	0.0512

H.3 Execution Time

	Direct Mode			**Split Mode**		
	YAML	Excel	Text	YAML	Excel	Text
Zero-shot						
Mean time (s)	**199**	227	204	390	410	382
One-shot						
Mean time (s)	100	107	**83**	244	223	196
CoT						
Mean time (s)	252	206	**183**	542	423	501
OS-CoT						
Mean time (s)	56	**52**	57	237	213	172

Table H.3: Mean execution time (s) per configuration