Master's Thesis

INTEROPERABILITY SIMULATOR FOR DATA SPACES

Aldi Doanta Kurnia

Computer Science - Software Technology

Graduation Committee:

dr. L. Ferreira Pires

dr. J.L. Rebelo Moreira

dr. M.A. Lopuhaä - Zwakenberg

D.R. Firdausy, EngD

Faculty of Electrical Engineering,
Mathematics and Computer Science

October 2023

UNIVERSITY OF TWENTE.

ACKNOWLEDGEMENT

I would like to express my gratitude to dr. L. Ferreira Pires, dr. J.L. Rebelo Moreira, and dr. M.A. Lopuhaä – Zwakenberg for their guidance, critical feedback, and positive support through this Master’s Project. Special thanks for D.R. Firdausy, EngD as my daily supervisor who introduced me to the topics of this project, for his constant support and for helping me set the right direction for this project. I would also like to thank the experts that gave me valuable feedback and suggestions during the prototype validation phase, and the Ministry of Communications and Informatics of the Republic of Indonesia for providing me with a scholarship to pursue my Master’s degree at the University of Twente.

Studying abroad and adapting to a new environment is not an easy task for me. Therefore, I would like to thank the communities I belong to during my studies in the Netherlands. I am grateful to my Indonesian friends from Vriendenkring (Yesaya, Raka, Michael, Silvi, Afif, Irine) and Karang Taruna Enschede, for the adventures, the delicious food, and the laughter we shared together. I am grateful to my brothers and sisters at the International Christian Fellowship Enschede for being a home away from home, especially to Yesaya, Raka, Michael, Daniel, Yosua, Juwita, Shanti, Stephen, Marria, Nick, and Rebekah. I am also grateful to my brothers and sisters at the Indonesian Christian Fellowship for the opportunity to share stories and grow together in Christ.

Obtaining my Master’s degree is the result of a process that spans seven years full of struggles and surprises. Therefore, I would like to express my gratitude to my childhood friends (Villy, Philip, Wilson) for more than 18 years of friendship, for supporting me in their own way through the moments spent together. I would also like to express my profound gratitude to the Ginting, Sembiring, and Tarigan families for their unwavering emotional support, encouragement, and prayers. Thank you for supporting my life decisions despite my stubbornness and my (sometimes naïve) idealism.

Finally, words cannot express my gratitude to the Almighty God, for helping me celebrate my failures, for giving me strength to live in the present moment, and for providing me assurance about what the future holds for me.

Soli Deo gloria.

Enschede, 20 October 2023

Aldi Doanta Kurnia

ABSTRACT

Many modern organizations need to share meaningful data with each other to improve their business operations. Data interoperability is the main prerequisite to achieve this goal. However, each organization might have different data standards and schemas. This situation is what motivates the creation of data space, a data-sharing network that enables its participants to exchange data regardless of data formats and schemas. International Data Spaces (IDS) is an example of a data space that prescribes a set of guidelines to make data exchange efforts easier.

As a data space, IDS only facilitates the data exchange activities. It does not provide a way to solve interoperability issues that might happen between two data participants, such as different names in the data schemas and unknown total costs for data access. We propose Interoperability Simulator as an additional IDS component to solve interoperability issues at the syntactic level. We designed the architecture and the business processes of the Interoperability Simulator, then analyzed how they can fit into the existing IDS architecture and business processes. The design artifacts were used to build a prototype that consists of two main functions, namely the Schema Matching function and the Pricing Calculation function.

We formulated three interoperability scenarios to observe the behavior of the prototype. We also gathered opinions from several experts to validate the design and prototype of the Interoperability Simulator. According to the validation results, we concluded that the current prototype of the Interoperability Simulator can be used to discover initial interoperability issues at the syntactic level. We also identified the limitations of the current prototype and proposed several points that can be carried out as future work.

Keywords
data interoperability, syntactic interoperability, schema matching, data spaces

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API	Application Programming Interface
BPMN	Business Process Model and Notation
CI	Continuous Integration
CSV	Comma Separated Values
DSSP	DataSpace Support Platform
ERD	Entity Relationship Diagram
GPT	Generative Pre-trained Transformers
HTTP	HyperText Transfer Protocol
ID	IDentifier
IDS	International Data Spaces
IDSA	International Data Spaces Association
JSON	JavaScript Object Notation
LAN	Local Area Network
OTM	Open Trip Model
RDBMS	Relational DataBase Management System
REST	REpresentational State Transfer
TNO	*Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek* (Netherlands Organization for Applied Scientific Research)
URL	Uniform Resource Locator
UUID	Universally Unique IDentifier
XML	eXtensible Markup Language
XSD	XML Schema Definition
YAML	YAML Ain't Markup Language

1. INTRODUCTION

Improvements in storage, communication, and computing technology have enabled organizations to utilize big data. In order to optimize its business operations, an organization needs to use data from the other organizations it interacts with. For example, if a supermarket wants to automatically restock its inventory under certain conditions, it needs to be able to send the purchase request automatically to the wholesaler. To make this process possible, the inventory levels owned by the supermarket can be processed by the wholesaler, which is a different organization. Moreover, the supermarket also needs to check the order status by accessing data provided by the wholesaler. This ability in which different organizations can exchange data between each other in a meaningful way is called *data interoperability*.

In practice, data interoperability is not always trivial to achieve. Each organization might have different conditions that make its data incompatible with data from the other organizations. For example, the American retail company Target failed to expand its business to Canada because of differences in measurement units in its systems (Gewirtz, 2016). To make data interoperable between organizations, they need to make agreements about their data exchange practices. The interacting organizations can agree to use formal standards published by standards organizations. Otherwise, they can define their own data exchange rules that are written in technical documentations. Regardless of the approach, these interacting organizations should form a data-sharing environment where the participants agree to use a common data exchange practice.

There are several approaches to create a data-sharing environment that enables data interoperability for organizations. One of them is the data space, which is a decentralized platform where qualified data participants can interact and exchange data between each other. The most prominent example of a data space platform is the International Data Spaces (IDS) by the International Data Spaces Association (IDSA) (IDSA, n.d.). IDS is a network of interfaces that enables data exchange between participating organizations, where these interfaces serve as endpoints to access the actual data held by the participating organizations. IDS uses a decentralized approach, in which there is no single entity that hosts and controls the entire data in a data space. However, those data space participants agree with a set of standards that enable them to seamlessly share and exchange data with each other. In a data space environment, the organizations that are eligible to join the data space should be able to communicate easily because of the agreed rules of the data space.

1.1 PROBLEM STATEMENT

In a data space environment, a typical data interoperability scenario involves an organization as a Data Consumer that needs to consume data from another organization as a Data Provider. The data exchange process might not happen smoothly because of the incompatibilities between the Data Provider and the Data Consumer. For example, the Data Consumer might expect the data to have attributes named *First Name* and *Last Name*, while the data provided by the Data Provider only contains an attribute called *Name*. Another example is when the Data Provider and the Data Consumer use different units of measurement to describe objects in the case about Target Canada mentioned before. These incompatibilities are considered as the information gap in a data interoperability scenario. Identifying the information gap between a Data Provider and a Data Consumer is the first step towards solving a data interoperability scenario.

To resolve the identified information gap, both the Data Provider and the Data Consumer have to agree on the meaning of the exchanged data. For example, if the information gap is caused by the difference in units of measurement, both parties must choose one common unit to be used. Data that does not use this common unit must be converted before being sent to the

Data Consumer. After the information gap has been resolved, the data exchange process can be executed using the standardized components of the data space. In this state, those two parties have solved the data interoperability scenario between them.

The example above shows that each data interoperability scenario must be solved on a case-by-case basis. A solution for a particular data interoperability scenario might not be easily replicated to other data interoperability scenarios. Each data interoperability scenario might have different factors that affect the level of information gap and the costs of implementing the data interoperability solution. For example, a data interoperability scenario can be solved by simply converting the data attributes from uppercase to lowercase. Another scenario might have more complex treatment because the numeric data instances have different measurement unit and value range.

Given the case-by-case nature of data interoperability solutions, designing and implementing a data interoperability solution to solve a data interoperability scenario is feasible in small scale, where there are only two organizations that need to interoperate with each other. However, implementing data interoperability solutions might be difficult to scale when there are several organizations that need to interoperate with each other. Therefore, the ability to automate some parts of the data interoperability scenario is required to improve the efficiency of data integration efforts. For example, automatic simulation of a data interoperability scenario will help the data participants identify the data incompatibilities and predict the required costs to implement the scenario.

1.2 OBJECTIVES AND RESEARCH QUESTION

The objective of this research is to design, implement, and validate a prototype of an Interoperability Simulator that can identify the information gap in an interoperability scenario. For a given data interoperability scenario between a Data Provider and a Data Consumer, the Interoperability Simulator assesses the information gap that hinders the data exchange process. Both the Data Provider and the Data Consumer can view a report about the assessed information gap, which is used to decide the required steps to solve the interoperability scenario. The Interoperability Simulator is deployed as an extension of an existing IDS data space, extending the IDS components as described in the IDS Reference Architecture Model (IDSA, 2022).

To guide the research direction, a research question was formulated as follows:

“How can an Interoperability Simulator help the interacting organizations measure costs of resolving the information gap in a data interoperability scenario in data spaces?”

Based on the research questions, three research sub-questions were defined as follows:

RSQ1: What attributes can be gathered by an Interoperability Simulator to assess the information gap in a data interoperability scenario? What are the limitations?

RSQ2: Using the gathered attributes from the Data Provider and the Data Consumer, how does an Interoperability Simulator assess the information gap?

RSQ3: How does an Interoperability Simulator report the assessed information gap to the interacting organizations?

1.3 RESEARCH SCOPE

In this research we applied three main constraints to narrow down the research scope:

1. The Data Provider and the Data Consumer form a data interoperability scenario that exists inside a data-sharing environment. The data-sharing environment uses the data space approach, which adopts the IDS architecture.
2. The Interoperability Simulator identifies the information gap at the syntactic level, without considering the semantic level.
3. The logistics domain was chosen as case study. The terminologies, data standards, and data schemas used in the case study are related to the logistics domain.

Figure 1 visualizes the research scope as an intersection of different aspects of this research. The size of the shape does not indicate the importance or the complexity of each aspect.

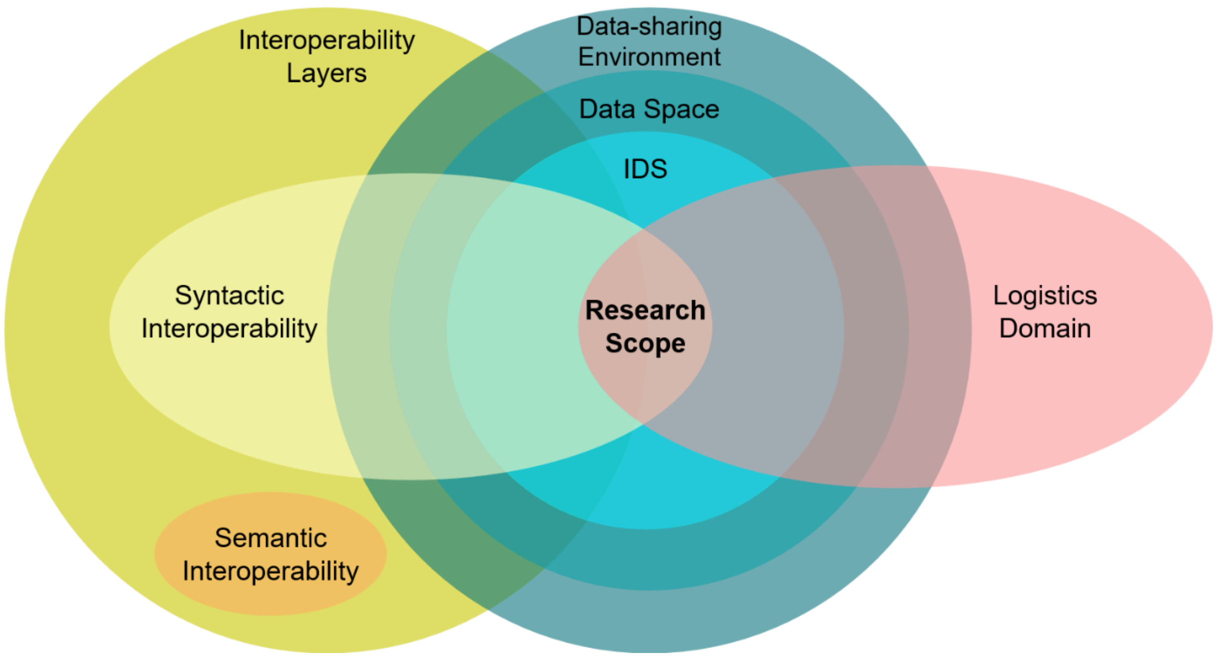


Figure 1 Visualization of the research scope

1.4 RESEARCH APPROACH

This research was performed in accordance with the Design Science Research Methodology by Wieringa (2014). The methodology is depicted in Figure 2 as a design cycle, consisting of three tasks:

1. **Problem Investigation**. The design cycle starts from this task, where the problem is studied, along with the related phenomena and the latest research related to the problem. Chapter 1 and Chapter 2 show the results of the Problem Investigation task of this research, describing the problem statement, the goals, and the background knowledge of the problem.
2. **Treatment Design**. This task is the next step after the Problem Investigation task, where the requirements and the design of the treatment are formulated. Chapter 3 reports on the results of the Treatment Design task, where the architecture design and the technical design of the treatment are explained.

3. **Treatment Validation**. This is the last task of the design cycle, where the treatment design is validated by developing a treatment prototype. The prototype validation process is also included in this task. The results of the Treatment Validation task are discussed in Chapter 4, Chapter 5, and Chapter 6.

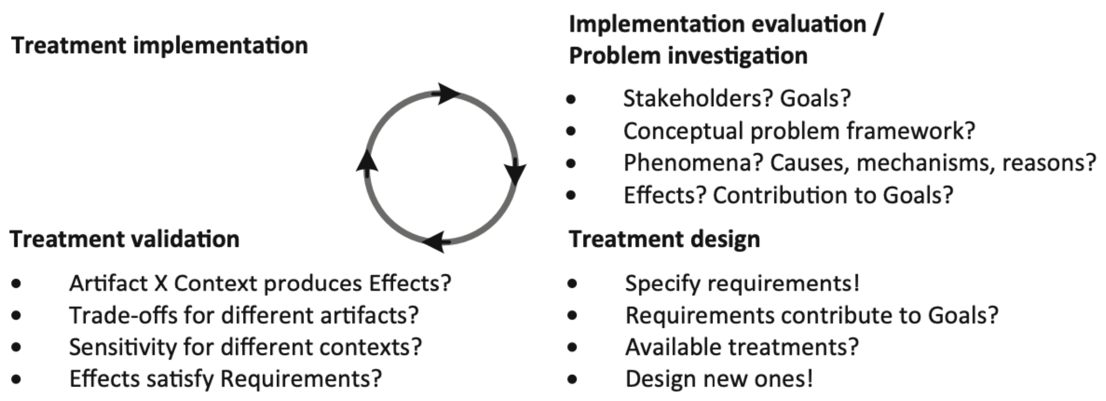

Treatment implementation
Implementation evaluation / Problem investigation
Stakeholders? Goals?
Conceptual problem framework?
Phenomena? Causes, mechanisms, reasons?
Effects? Contribution to Goals?
Treatment validation
Artifact X Context produces Effects?
Trade-offs for different artifacts?
Sensitivity for different contexts?
Effects satisfy Requirements?
Treatment design
Specify requirements!
Requirements contribute to Goals?
Available treatments?
Design new ones!

Figure 2 The three tasks of the design cycle prescribed by Design Science Research Methodology (excluding the Treatment Implementation task), as a part of the engineering cycle (Wieringa, 2014)

The Treatment Implementation task and the Implementation Evaluation task are outside the scope of the research, since they are part of a larger engineering cycle. However, these two tasks could be performed in the future research if several logistic companies are willing to treat a real-life problem using the designed artifacts.

1.5 STRUCTURE

This thesis is further organized as follows. Chapter 2 provides the background knowledge about data interoperability and data space. Chapter 3 describes the proposed treatment design for both the data space and the Interoperability Simulator. The details about the prototype development are described in Chapter 4, while Chapter 5 and Chapter 6 describe the interoperability scenarios and the prototype validation, respectively. Chapter 7 concludes the thesis by giving our conclusions, research limitations, and topics for future work.

2. BACKGROUND

This chapter explains several background concepts related to the Interoperability Simulator and data space. Section 2.1 describes the definition of data interoperability and the overview of the data space concept, while Section 2.2 introduces IDS as an example of data space. Section 2.3 and Section 2.4 describe the two IDS layers that are relevant to this research, namely the System Layer and the Process Layer. Section 2.5 explains the concept of information gap, while schema matching as the solution to solve the information gap is discussed in Section 2.6.

2.1 DATA INTEROPERABILITY AND DATA SPACE

Several definitions of interoperability can be found in two literature sources: IEEE (1991) defines interoperability as the “ability of multiple systems or components to exchange information and to use the exchanged information”, while Wegner (1996) defines interoperability as the “ability of multiple software components to cooperate despite having different languages, interfaces, and execution platforms”. Wegner gives more emphasis on interoperability in the client-server paradigm. Those definitions agree in that the main keywords that define interoperability are “multiple different systems” and “information exchange”.

These broad definitions of interoperability can be made more specific when the term is given in a more specific context. For example, the term “data interoperability” refers to the ways of formatting data that allow diverse datasets to be processed together in meaningful ways (Network of the National Library of Medicine, n.d.). This definition is consistent with the definitions of interoperability that were mentioned before. The term “diverse datasets” corresponds to “multiple different systems”, while the term “datasets that are processed together” corresponds to “information exchange”.

The term *data space* was introduced by Franklin et al. (2005), in which they describe the data space concept as a novel approach of data management, which usually uses a centralized Relational Database Management System (RDBMS). Data space is considered as a data co-existence approach, where the data space accommodates all kinds of data sources regardless of how integrated they are. To support these diverse data sources, a data space is equipped with a DataSpace Support Platform (DSSP) that provides basic services such as data catalogue and data search.

Halevy et al. (2006) further discussed the data space idea that was proposed by Franklin et al. (2005). They reiterated the differences between DSSP and traditional databases and data integration systems, giving emphasis on how the data space approach has less upfront costs compared to the conventional data integration system. These reduced upfront costs are possible because DSSP does not require full semantic integration from the data sources that join the data space ecosystem. The high-effort data integration process can be postponed until absolutely needed, such as when two data participants need to exchange data.

2.2 INTERNATIONAL DATA SPACES (IDS) OVERVIEW

IDS is an example of DSSP that is advocated by the International Data Spaces Association (IDSA). According to IDSA (2022), IDS strives to fulfil several strategic requirements:

1. IDS must be secure and trusted. Each data space participant must pass a certification procedure before being allowed to join the ecosystem.
2. Data sovereignty must be achieved in IDS, where each individual or organization is able to determine what can be done with its own data.

3. IDS must be built as a decentralized data ecosystem, where the original data remains with the respective data owner until the data exchange happens with the data consumer. Data economy should emerge from this data ecosystem, where new business models related to data services can be utilized by the data owners.
4. IDS must have standardized interoperability, allowing meaningful data exchange between different organizations with different data interfaces.

According to IDSA (2022), the five layers of the IDS Reference Architecture Model are:

1. **Business Layer**, which gives an abstract description of the roles involved in IDS.
2. **Functional Layer**, which defines the functional requirements that must be provided by IDS.
3. **Information Layer**, which describes the information model that represents the concepts and relationships that are related to IDS.
4. **Process Layer**, which explains how the IDS components interact with each other.
5. **System Layer**, which specifies the technical details of the IDS components.

The System Layer and the Process Layer are closely related to the technical aspects of data interoperability. These two layers are discussed in the following sections.

2.3 IDS SYSTEM LAYER

The System Layer is the technical layer that specifies the technical details of IDS. It translates the definitions from the higher-level layers such as Business Layers and Process Layers to a concrete data and services architecture.

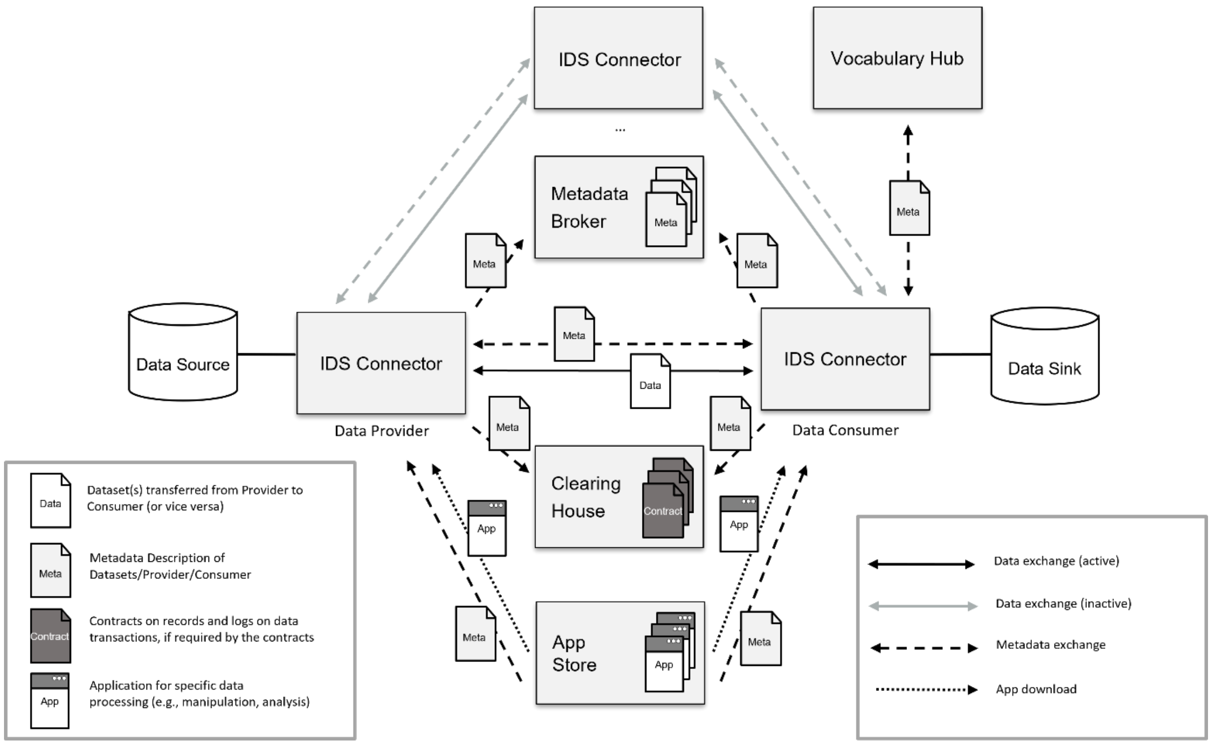

IDS Connector
Vocabulary Hub
...
Metadata Broker
Meta
Data Source
IDS Connector
Data Provider
Data
IDS Connector
Data Consumer
Data Sink
Clearing House
Contract
App
App Store
Dataset(s) transferred from Provider to Consumer (or vice versa)
Metadata Description of Datasets/Provider/Consumer
Contracts on records and logs on data transactions, if required by the contracts
Application for specific data processing (e.g., manipulation, analysis)
Data exchange (active)
Data exchange (inactive)
Metadata exchange
App download

Figure 3 Overview of the IDS architecture, illustrating the relationship between data provider and data consumer in a data space (IDSA, 2022)

Figure 3 shows the architectural overview of IDS along with its main components. It illustrates a scenario where a Data Consumer communicates with a Data Provider. Both Data Provider and Data Consumer are participants of a data space. Each data space participant has an IDS Connector that acts as an interface between the data space and the participants. Before data exchange occurs between the two parties, Data Consumer must agree with the usage policies set by the data owner. These policies can vary for each data owner, ranging from rules about data access to the payment model for commercial data.

After the usage policies has been agreed, data can be transferred to the Data Consumer. Data apps that are installed to the IDS Connector can be used to perform additional tasks with the data, such as data transformation, data aggregation, or data analysis. These data apps can be developed in-house or obtained from the App Store of the data space.

Some IDS components are described below.

2.3.1 IDS Connector

The IDS Connector is an interface that enables an organization to interact with other organizations in a data space. IDS Connectors are the main entities that form a network in a data space, similar to electronic devices that form a network in a Local Area Network (LAN). An IDS Connector is deployed as a group of containers, where each container offers a runtime environment to an application that executes a specific function. Figure 4 illustrates the architecture of an IDS Connector.

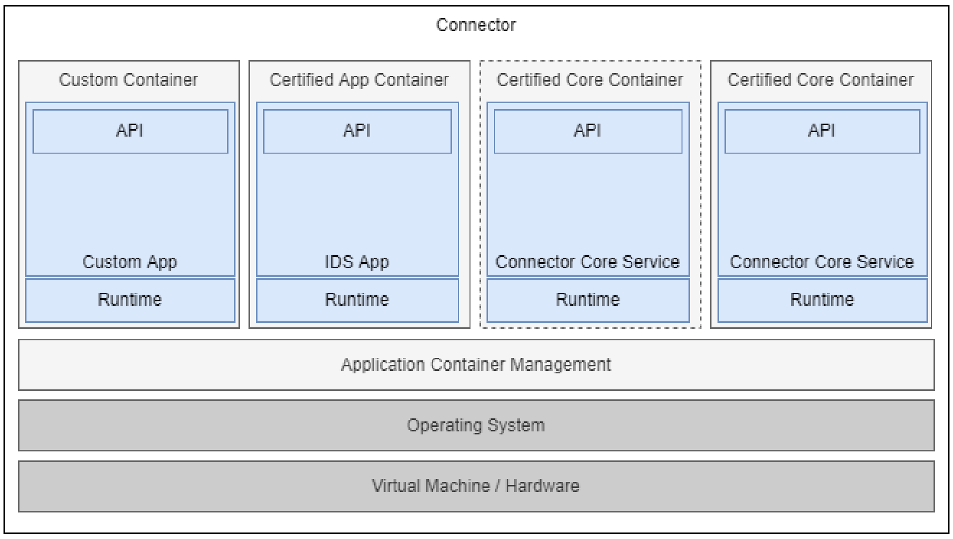


Figure 4 IDS Connector architecture (IDSA, 2022)

An IDS Connector consists of these components:

1. Containers, where each container has a specialized functionality. There are a few different types of containers in an IDS Connector:

    a. Certified Core Container. This container type is related to the main functionalities of an IDS Connector, such as data management, configuration management, and IDS Protocol Authentication. Each container is responsible to run a specific functionality. Therefore, there are multiple Certified Core Containers depending on the functionalities supported by the IDS Connector.

    b. Certified App Container. This is a certified container downloaded from the App Store that runs a specific IDS App for the IDS Connector.

    c. Custom Container. Most containers of an IDS Connector are certified by IDS to ensure quality standards. However, some containers run Custom Apps that were developed by organizations. These containers are for internal use and usually require no certification.

2. Application Container Management. For scalability reasons, some containers need to be automatically scaled depending on the workload. These containers also need to be able to restart automatically whenever they fail. Therefore, Application Container Management is needed to manage the container resources in an IDS Connector. Application Container Management is optional and might not be necessary for non-production environments.

3. Operating System that runs the container manager and the containers, along with the virtual machine or hardware the Operating System is running on.

2.3.2 Vocabulary Hub

We recall that one of the goals of IDS is to achieve standardized interoperability. To achieve this goal, it is necessary for a data space to use common terms to describe data models and services. A collection of these standard terms is called a vocabulary, which is shared by every participant in a data space. Moreover, the vocabulary can be enriched with domain-specific terms that are used by the data space (IDSA, 2022). This vocabulary is stored in a Vocabulary Hub that is accessible by the IDS Connectors.

2.3.3 IDS Apps

IDS Apps are reusable software applications that can be deployed and executed on an IDS Connector. Each IDS App has a specific functionality that complements the core functionalities of an IDS Connector. These applications are developed by the IDS participants and have passed certification criteria set by IDS. Different types of IDS can be bundled to create a data processing pipeline. The distribution of IDS Apps is managed by the IDS App Store (IDSA, 2022), whose architecture is depicted in Figure 5.

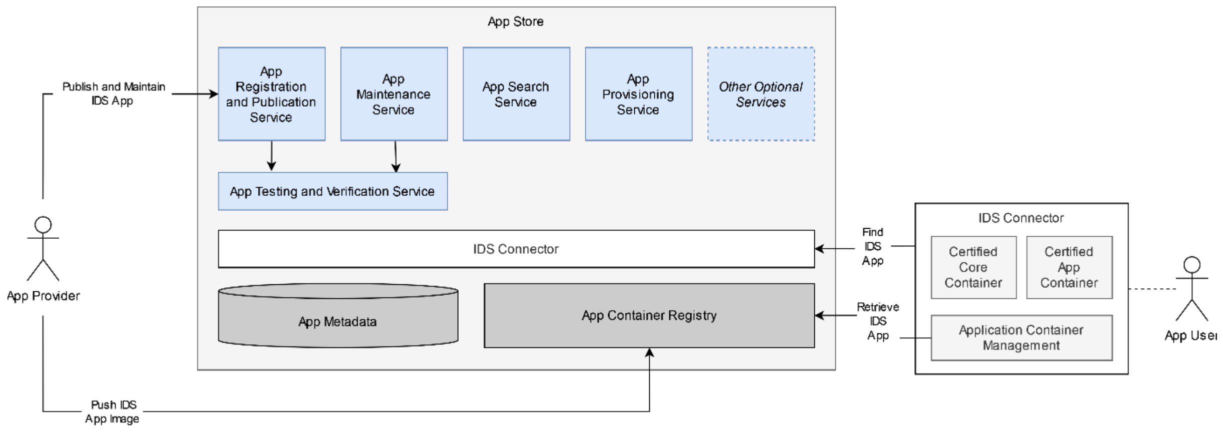


Figure 5 App Store architecture (IDSA, 2022)

2.4 IDS PROCESS LAYER

The Process Layer describes how an IDS component interacts with the other IDS components. In some domains, the Process Layer is also called the Business Process Layer. The Process Layer of IDS consists of the following steps:

1. Onboarding: what to do to be granted access to the International Data Spaces as a Data Provider or Data Consumer.

2. **Data Offering:** offering data or searching for a suitable data.

3. Contract Negotiation: accept data offers by negotiating the usage policies.

4. Exchanging Data: transfer data between IDS Participants.
5. **Publishing and using IDS Apps:** interacting with an IDS App Store or using IDS Data Apps.

The two steps related to the relationship between IDS Connector and Data Apps are explained further below.

2.4.1 Data Offering

In typical use cases of data spaces, a Data Provider does not know the other participants that is interested in the Data Provider's data. Therefore, a Data Provider needs to give a meaningful description and other metadata about its data (IDSA, 2022). When a Data Provider wants to publish data to the data space, it can create a Self-Description of its data asset. A Self-Description of a data asset contains metadata such as the description, the license, the language, and the media type of the data asset. Domain-specific vocabulary can also be added to the Self-Description. After that, the Self-Description is deployed at the IDS Connector of the Data Provider.

To increase visibility of the IDS Connector, the Self-Description can be published to the IDS Metadata Broker, while the vocabulary can be published to the Vocabulary Hub and linked to the Self-Description. This way, the Data Provider becomes accessible to the other IDS Connectors (IDSA, 2022).

On the other hand, a Data Consumer also needs to look for a suitable Data Provider. The Data Consumer can either use the catalogs of the IDS Metadata Broker to search for metadata, or crawl the self-descriptions that is embedded in IDS Connectors. The business processes of registering Self-Description and Vocabulary are depicted in Figure 6 and Figure 7 using Business Process Model and Notation (BPMN).

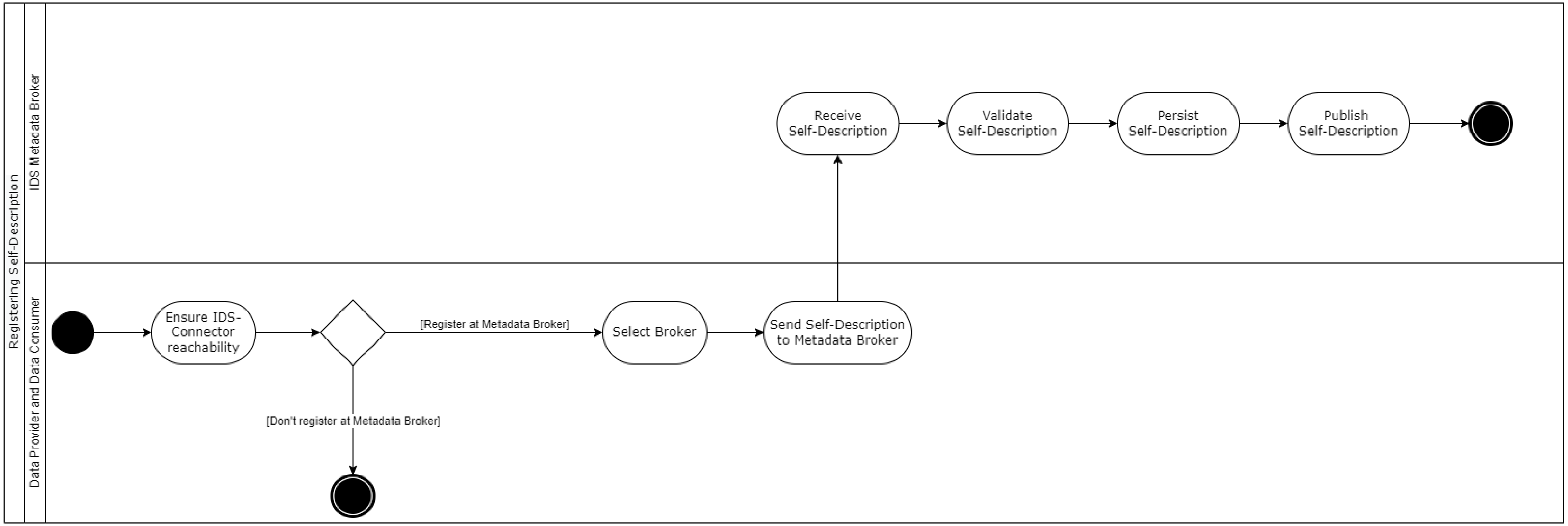

Registering Self-Description
IDS Metadata Broker
Data Provider and Data Consumer
Ensure IDS-Connector reachability
[Register at Metadata Broker]
[Don't register at Metadata Broker]
Select Broker
Send Self-Description to Metadata Broker
Receive Self-Description
Validate Self-Description
Persist Self-Description
Publish Self-Description

Figure 6 BPMN diagram of the "Register Self-Description at Metadata Broker" business process (IDSA, 2022)

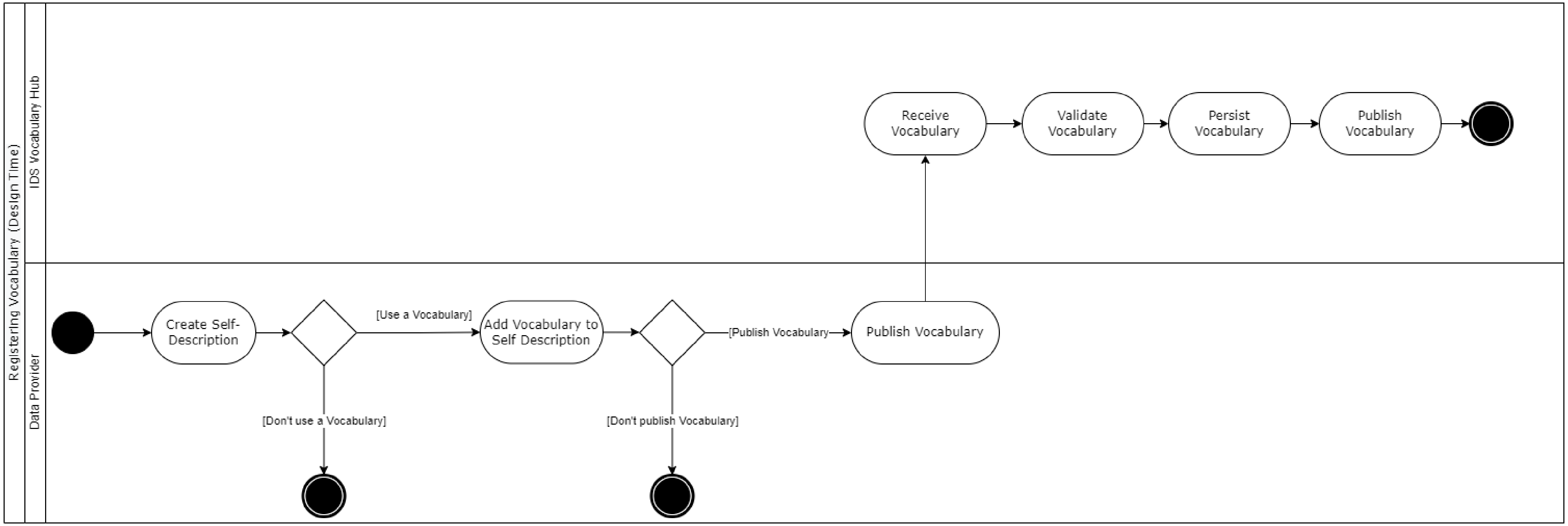


Figure 7 BPMN diagram of the "Register Vocabulary at Vocabulary Hub" business process (IDSA, 2022)

2.4.2 Publishing and Using IDS Apps

IDS Apps are software components that can be installed on an IDS Connector to perform certain data processing tasks. An App Provider can create IDS Apps and publish them on the IDS App Store, making them available for all the data space participants. As part of the quality assurance process, some IDS Apps require certification as a prerequisite for publishing. Regardless of the certification requirement, publishing an IDS App requires the App Provider to publish both the app image and the app metadata. The app image is published to the App Store's Container Registry, while the app metadata is published to the App Store's database (IDSA, 2022).

An App User (such as a Data Consumer) that needs to use IDS Apps can perform a search query for the apps in the IDS App Store. The App Store then shows a list of relevant IDS Apps, along with the relevant metadata. Some IDS App Providers might require the App User to pay for the selected IDS Apps. Finally, the App User retrieves the IDS Apps and deploys them to the App User's IDS Connector. The business process of using an IDS App is shown in Figure 8.

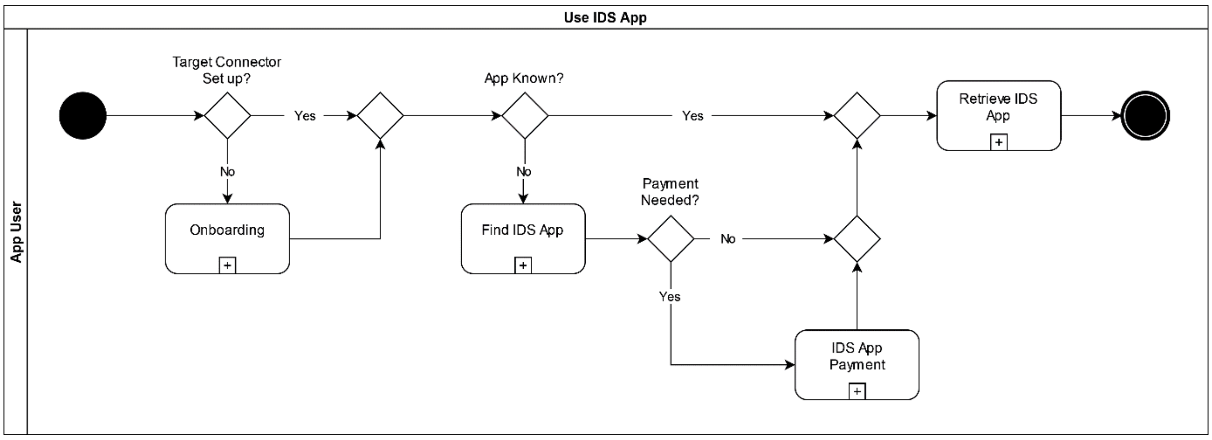


Figure 8 BPMN diagram of the "Use IDS App" process (IDSA, 2022)

2.5 INFORMATION GAP

We recall that data exchange might not always happen immediately because the provided data by the Data Provider is not compatible with the requested data by the Data Consumer. There are different terms that can be used to define this situation, such as *data heterogeneity* (Sheth, 1998; Sheth & Kashyap, 1992) and *data incompatibility* (Nagarajan et al., 2007). However, we use the term *information gap* as an umbrella term to describe the differences between the provided data and the requested data.

There are at least two approaches to solve the information gap problem. The first approach uses custom rules or mappings to transform the provided data into the requested data, while the second approach involves creating mappings to a generic domain model and using it to transform the data (Nagarajan et al., 2007). Both approaches require creation of rules and mappings to transform the data into another format that is acceptable by the Data Consumer.

In an interoperability scenario that involves two parties, Nagarajan et al. (2007) classified data incompatibilities into three categories:

1. **Attribute-level incompatibilities**, which occur when semantically similar attributes are described using different descriptions.

2. **Entity-level incompatibilities**, which occur when semantically similar entities are described using different descriptions, such as different number of attributes and different type of attributes.
3. **Abstraction-level incompatibilities**, which are a mix of attribute-level and entity-level incompatibilities. They occur when semantically similar attributes and/or entities are described at different levels of abstraction.

Table 1 shows the different types of data incompatibility with a few examples for each incompatibility type, where the conflicting attributes/entities are highlighted in red.

Table 1 Categorization of data incompatibilities (Nagarajan et al., 2007)

Incompatibilities	Provided Data	Requested Data
Attribute-level Incompatibilities		
Naming conflict Semantically similar attributes have different names (synonyms).	Student (Id, Name)	Student (SSN, Name)
Semantically unrelated attributes have the same name (homonyms).	Student (Id, Name)	Book (Id, Name)
Data representation conflict Semantically similar attributes have different data types or representations.	Student (Id, Name) Id is defined as a 4-digit number.	Student (Id, Name) Id is defined as a 9-digit number.
Data scaling conflict Semantically similar attributes are represented using different scales.	Score 1-100	Score A-F
Entity-level Incompatibilities		
Naming conflict Semantically similar entities have different names (synonyms).	Employee (Id, Name)	Worker (Id, Name)
Semantically different entities have the same name (homonyms).	Ticket (Id, MovieId, MovieName)	Ticket (Id, FlightId, Departure, Arrival)
Schema isomorphism conflict Semantically similar entities have different structure because of different number and/or type of attributes.	Person (Name, Address, HomePhone, WorkPhone)	Person (First Name, Last Name, Address, Phone)
Abstraction-level Incompatibilities		
Generalization conflict Semantically similar entities are represented at different levels of generalization.	Grad-Student (Id, Name, Major)	Student (Id, Name, Major, Type) Entity Student is a more general form of entity Grad-Student.
Aggregation conflict Semantically similar entities where one entity is represented as an aggregate of the other entity.	Professor (Id, Name, Dept)	Faculty (Id, ProfId, Dept) Entity Faculty is an aggregate entity of multiple Professor entities.

Attribute entity conflict A semantically similar entity is represented as an entity in one version and as an attribute in the other version.	Course (Id, Name, Quartile)	Dept (Id, Course, Quartile)

2.6 SCHEMA MATCHING

A schema is a description of the logical structure of a database (IEEE, 1991). Even though this early definition of *schema* associates the term with relational databases, the term *schema* has been being used to describe a structure of any kind of data object. A schema represents the information about a data object's structure, such as attribute names, attribute types, and their value constraints. For example, the JSON schema of the trip data in the logistics domain can contain attribute names such as `trip_status`, `transport_mode`, and `destination_coordinate`, where each attribute has its own data type and constraints.

Schema matching is a process of detecting similarities of elements between multiple schemas and find a mapping among their elements. Figure 9 shows a simple schema matching scenario that aims to find a mapping between the elements of a table schema with the elements of another table schema (Ionescu, 2020). Rahm & Bernstein (2001) classifies the schema matching approaches, which are also called matchers, into two different categories, namely individual matchers and combined matchers. The individual matchers use a single matching criterion to match the schemas, while the combined matchers use a combination of different individual matchers to perform the schema-matching task. The individual matchers are further classified into two categories based on the representation of the input.

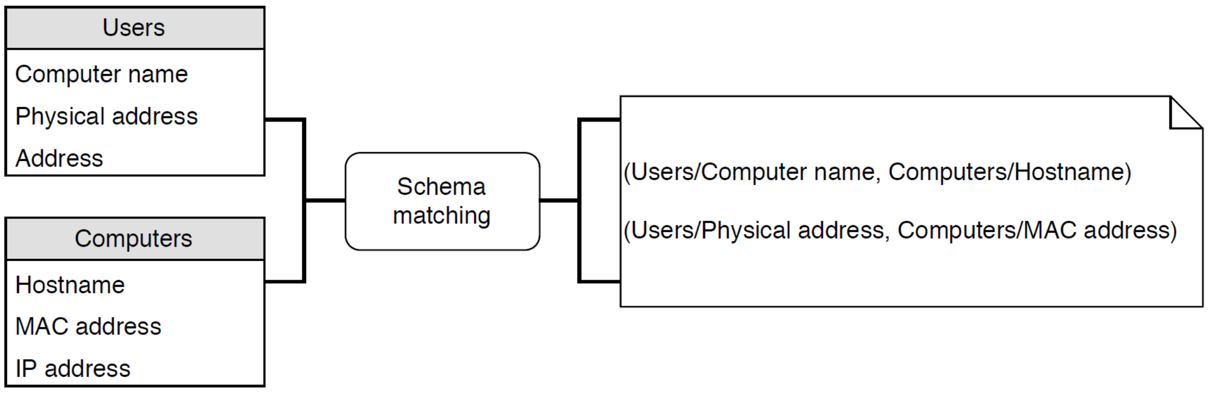


Figure 9 Schema matching that takes two table schemas as input, producing a list of pairs that represent their attribute mappings (Ionescu, 2020)

2.6.1 Schema-level Individual Matchers

Schema-level matchers rely on the information contained in the schema, without using the information contained in the instance data. Schema information usually consists of properties of schema elements, such as name, data type, type constraints, and schema structure. Depending on the match granularity, schema-level matchers can be further categorized into two matching approaches (Rahm & Bernstein, 2001):

1. **Element-level matching**, which is a matching approach that only considers the individual schema elements, such as columns in a database schema or properties in a JSON schema. The examples of element-level matching are matching database columns based on name similarity or type similarity.

2. **Structure-level matching**, which is a matching approach that considers the combination of schema elements that form the schema structure. The approach can use graphs or trees to model the structure of the schemas, then compare the structures to find appropriate mappings.

2.6.2 Instance-level Individual Matchers

Instance-level matchers use the data instances of the schemas to perform the schema matching process. For example, instance-level matchers can match two schema elements based on the similarity of their data instances. Some element-level matchings that are used in schema-level matchers can be used in instance-level matchers, because some element-level matchings can be applied to data instances.

2.6.3 Combined Matchers

The individual matchers explained above can be combined to improve the accuracy of the schema mapping. According to Rahm and Bernstein (2001), there are two different types of matchers:

1. **Hybrid matcher**, which combines different matching criteria to generate a single mapping result. Each matching criterion can be handled by a specific schema-matching algorithm, generating a partial mapping that can be combined with partial mappings from the other schema-matching algorithms.
2. **Composite matcher**, which combines the results of independently executed schema-matching matchers. The schema-matching matchers can be performed either sequentially or simultaneously. If they are performed in a sequential way, the mapping results from a schema-matching algorithm are used as input parameters for the next schema-matching algorithm. One of the examples of the composite matcher is COMA, a generic schema-level matcher developed at the University of Leipzig (Do & Rahm, 2002).

2.6.4 Some Matcher Approaches

There are different types of matchers that apply different schema-matching strategies. This section gives three examples of matcher that are relevant to this research.

COMA

COMA is an ontology matching tool and a composite matcher that comprises several schema matching strategies. A schema in COMA is represented as a rooted directed acyclic graph, while its schema elements are represented as graph nodes that might have referential relationships with the other schema elements (Do & Rahm, 2002).

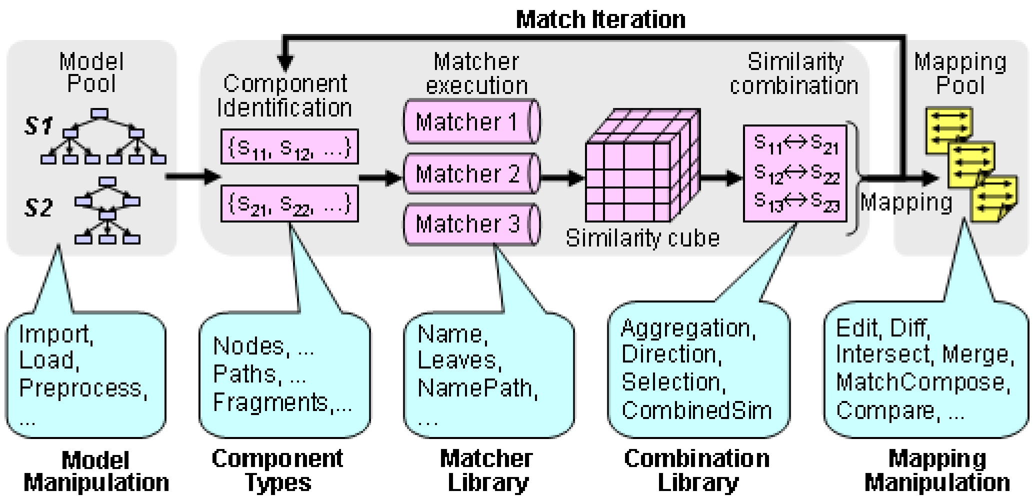


Figure 10 Match processing in COMA 3.0 (Database Group Leipzig, n.d.)

Figure 10 shows the overview of match processing in COMA 3.0, which is the latest version of COMA. The initial step starts with importing the schema pairs `S1` and `S2` and creating a graph representation for each schema. The graphs are used in the match iteration loop, in which the schema elements are identified and passed as input to the individual matchers. Every schema matching operation that consists of a matching strategy, an element from `S1`, and an element from `S2` is then stored in a similarity cube. The results of schema matching operations in the similarity club are combined at the end of the match iteration loop. The matching process can be repeated using different matching strategies to improve the mapping results. Otherwise, the resulting mapping is returned as the final mapping.

Cupid

Cupid is a schema-level individual matcher that uses a tree representation to model a schema. The schema elements and their hierarchical relationships are modeled as several interconnected tree nodes. The tree nodes are used to calculate the similarity coefficients between the schema elements. These similarity coefficients are used to formulate the final schema matching result. According to Madhavan et al. (2001), the schema matching process in Cupid is divided into three phases:

1. **Linguistic matching phase**, in which linguistic approaches are applied to match the schema elements. For example, the schema elements can be matched based on the name similarity, the data types, or the domain context of the term. A thesaurus is employed to help Cupid identify abbreviations and synonyms. This phase produces a linguistic similarity coefficient between each element pair.
2. **Structural matching phase**, in which the schema elements are matched based on the similarity of the schema structure. A schema structure contains information about the context of the schema elements and their neighbors. This phase gives an output of structural similarity coefficient for each element pair.
3. **Mapping generation phase**, in which a mapping is generated by selecting schema element pairs based on a formula of weighted similarity.

Similarity Flooding

Similarity Flooding is a matcher that utilizes graphs to model data schemas and data instances. Similarity Flooding performs the schema matching operation that consists of four sequential steps (Melnik et al., 2002):

1. Schemas transformation from their native representation into directed labeled graphs. The schema elements are modeled as graph nodes. The transformation process uses an import filter that preserves the definitions of the relational schemas.
2. Creation of the initial mapping between the two schema graphs. The initial mapping is created using a string matcher that compares prefixes and suffixes of literals in the schemas.
3. An execution of the iterative Similarity Flooding algorithm, producing a similarity propagation graph that is derived from the schema graphs. The Similarity Flooding algorithm is based on an assumption that each pair of schema elements (which are modeled as graph nodes) propagates its similarity to its neighbors. The iterative propagation process will eventually converge at one point, resulting in the final similarity score.
4. A filtering step, in which a subset of node pairs is selected as the best matching results according to the algorithm.

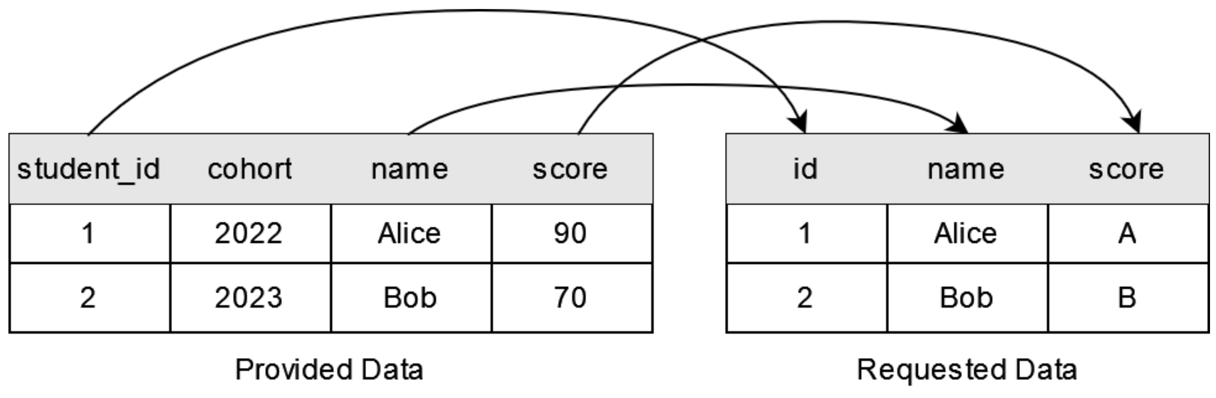
3. SOLUTION DESIGN

As described in Section 2.2, IDS strives to follow four strategic requirements to enable a decentralized data ecosystem. One of these strategic requirements is standardized interoperability, where data participants can communicate and exchange data with each other using a set of agreed standards. The Interoperability Simulator is one of the proposed solutions to realize standardized interoperability.

This chapter reports on the Treatment Design task of the design cycle, in which the solution design of the Interoperability Simulator is proposed and its relationship with the IDS data space is explained. The chapter starts with the definition of interoperability scenario in Section 3.1, followed by Section 3.2 that describes the software requirements specification of the Interoperability Simulator. Section 3.3 discusses the proposed reference architecture and how the solution would fit in the existing business processes of IDS. The proposed architecture and the revised business processes are based on the software requirements specification and our research scope defined in Section 1.3.

3.1 INTEROPERABILITY SCENARIO

We define an interoperability scenario as a situation where a Data Consumer requests a data resource from a Data Provider, where the data structure requested by the Data Consumer is possibly different from the data structure provided by the Data Provider. Figure 11 illustrates an example interoperability scenario with two different data structures as tabular data.



student_id	cohort	name	score
1	2022	Alice	90
2	2023	Bob	70

Provided Data

id	name	score
1	Alice	A
2	Bob	B

Requested Data

Figure 11 Illustration of an interoperability scenario. The arrows above the column names indicate the mapping from the provided data structure to the requested data structure.

In the example scenario above, three adjustments are needed to transform the Provided Data to the Requested Data:

1. Rename the `student_id` attribute to `id`.
2. Remove the `cohort` attribute.
3. Translate the data instances of `score` attribute from numeric scale to letter-based scale.

Interoperability scenarios are not explicitly mentioned in the IDS Reference Architecture Model. However, according to IDS Process Layer, it can be inferred that an interoperability scenario can happen during the Data Offering step, when the Data Consumer is searching for suitable data in the data space. During this step, the Data Consumer do not have access to the data yet, since the usage policies agreement has not been reached between the Data Provider and the Data Consumer. The Data Consumer must rely on metadata such as data description, file format, and data schema. Therefore, an interoperability scenario in IDS can only be solved using schema-level matchers, as explained in Section 2.6.1. As a consequence, step 3 in the previous example scenario cannot be performed, since the transformation of data instances can only happen after the Data Consumer gains access to the actual data.

Chapter 5 presents the examples of interoperability scenario that have been considered in this research.

3.2 SOFTWARE REQUIREMENTS

To achieve the IDS strategic goal of standardized interoperability, we formulated the software requirements of the Interoperability Simulator using the Goal-Design Scale approach by Lauesen (2002). Table 2 shows the software requirements of the Interoperability Simulator from different levels, ranging from the high-level business requirements to the application-specific requirements.

Table 2 Software requirements of the Interoperability Simulator

Reference	Requirement
Goal-Level Requirements: business requirements that the solution users want to achieve	
GR1	The Interoperability Simulator should ease the integration effort of data participants in a data space.
Domain-Level Requirements: support the solution users to achieve a particular task	
DoR1	The Interoperability Simulator should help the Data Consumer gather the required information to assess the information gap.
DoR2	The Interoperability Simulator should support the Data Consumer in assessing the information gap using the gathered information.
DoR3	The Interoperability Simulator should be able to report the assessed information gap to the Data Consumer.
DoR4	The Interoperability Simulator should help the Data Provider comply with the existing standard data models in a data space.
Product-Level Requirements: requirements about the functionalities of the solution	
PR1	The Interoperability Simulator should collect the provided schema from the Data Provider and the requested schema from the Data Consumer.
PR2	The Interoperability Simulator should collect the standard schema from the standard data model, which is managed by the Vocabulary Hub.
PR3	The Interoperability Simulator should collect the pricing information of the data offered by the Data Provider in case of commercial data.
PR4	The Interoperability Simulator should assess the information gap by comparing the schemas of the Data Provider, the Data Consumer, and the data space's standard data model.
PR5	Based on the pricing information from the Data Provider, the Interoperability Simulator should calculate the total costs of accessing the commercial data.
PR6	The Interoperability Simulator should report the assessed information gap and the total costs to the Data Consumer.
Design-Level Requirements: requirements that are related to the user interface of the solution	
DeR1	The Interoperability Simulator should enable the Data Consumer to create an interoperability scenario by selecting a data resource from a Data Provider.
DeR2	The Interoperability Simulator should show the assessed information gap and the possible costs to the Data Consumer.

The domain-level requirements are derived from the research sub-questions in Section 1.2. These requirements are the interoperability-related tasks that need to be accomplished by the stakeholders. There is only one type of stakeholder with two different roles, as mentioned in

Section 1.3. Therefore, there is at least one domain-level requirement for each role. The product-level requirements (indicated by PR1-PR6) are derived from the research sub-questions in Section 1.2, by replacing attributes by more specific terms such as schema and data space.

3.3 REFERENCE ARCHITECTURE AND BUSINESS PROCESSES

We defined a reference architecture based on the specified research scope given in Section 1.3 and the software requirements of Section 3.2. It serves as the template architecture of the Interoperability Simulator, as it is used to define a more specific and concrete architecture in Chapter 4. The reference architecture was defined using ArchiMate 3.2 language (The Open Group, 2022), and it models the Interoperability Simulator from three different architectural layers, namely the Business Layer, the Application Layer, and the Technology Layer.

The relationship between the three layers can be depicted by ArchiMate views, each representing a system from a particular perspective. Each view is defined in accordance with a viewpoint, which specifies the conventions for this particular kind of view (The Open Group, 2022). There are several standard viewpoints that can be used as guidelines, as well as custom viewpoints that use standard ArchiMate elements.

The reference architecture has been defined from two views, which are illustrated in Figure 12 and Figure 13. The components in yellow belong to the Business Layer, the components in blue belong to the Application Layer, and the components in green belong to the Technology Layer.

The components in white are considered out of scope of this research, and therefore will not be discussed in detail in the subsequent chapters. This also means that the functionalities indicated in these white components are not included in software requirements specification in Section 3.2. However, they are part of the complete vision of the Interoperability Simulator as an interoperability solution in a data space environment.

Each view is explained further in the sequel.

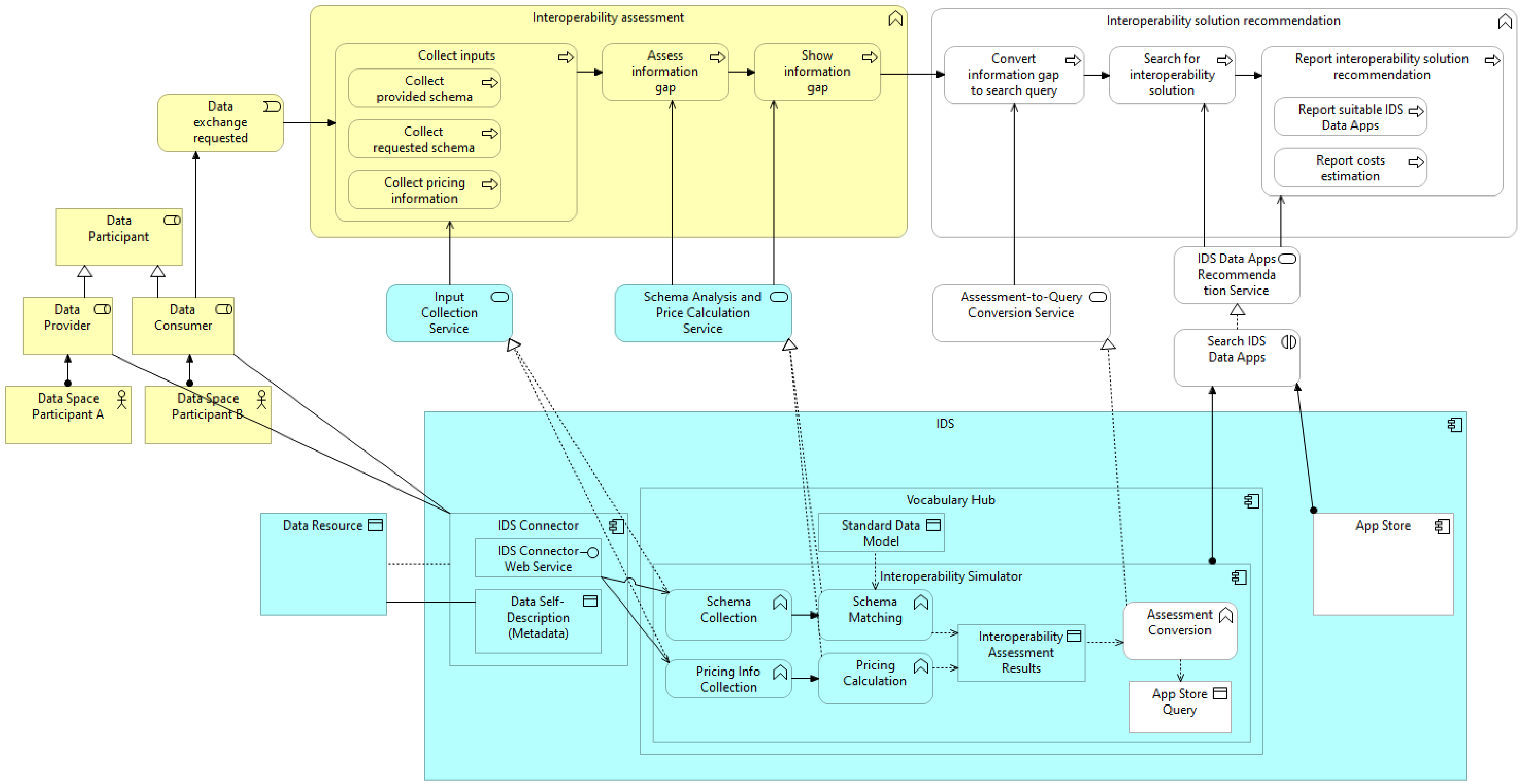

Interoperability assessment
Collect inputs
Collect provided schema
Collect requested schema
Collect pricing information
Assess information gap
Show information gap
Interoperability solution recommendation
Convert information gap to search query
Search for interoperability solution
Report interoperability solution recommendation
Report suitable IDS Data Apps
Report costs estimation
Data exchange requested
Data Participant
Data Provider
Data Consumer
Data Space Participant A
Data Space Participant B
Input Collection Service
Schema Analysis and Price Calculation Service
Assessment-to-Query Conversion Service
IDS Data Apps Recommendation Service
Search IDS Data Apps
IDS
Vocabulary Hub
Data Resource
IDS Connector
IDS Connector Web Service
Data Self-Description (Metadata)
Standard Data Model
Interoperability Simulator
Schema Collection
Schema Matching
Pricing Info Collection
Pricing Calculation
Interoperability Assessment Results
Assessment Conversion
App Store Query
App Store

Figure 12 Application Usage View

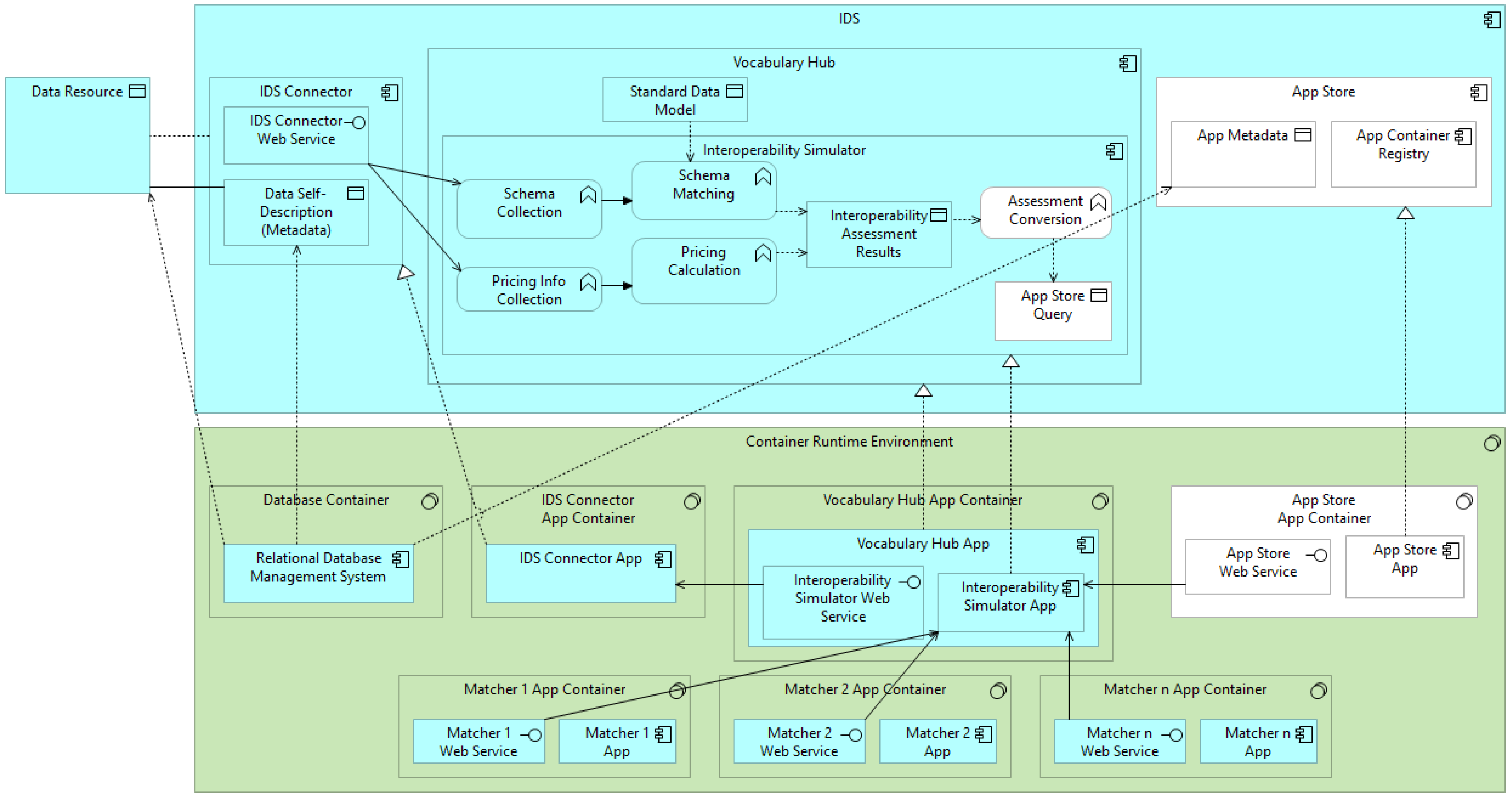


Figure 13 Implementation and Deployment View

3.3.1 Application Usage View

The Application Usage View depicted in Figure 12 shows the relationship between the Business Layer and the Application Layer. More specifically, it shows how the applications support the business processes, and how the applications interact with each other. The Application Usage View uses a basic ArchiMate viewpoint called the Application Usage Viewpoint (The Open Group, 2022).

The Data Space Participants are the only stakeholders who are relevant to the Interoperability Simulator. These participants are the individual or organizations that have interests to interact with each other in the data space. In an interoperability scenario, each Data Space Participant can assume a role as either a Data Provider or a Data Consumer. Regardless of the role, each Data Space Participant uses the IDS Connector application component to interact with the data space entities. The IDS Connector also serves as an interface between the Data Resource owned by Data Provider and the data space. The Data Resource is stored in Data Provider's private repository, preserving Data Provider's sovereignty over its own data.

The Business Layer of the Application Usage View consists of two main business functions:

1. A business function that groups several business processes that are related to interoperability assessment, which starts from the business process of collecting schemas from Data Participants and ends at the business process of showing the information gap to Data Consumer.
2. A business function that groups the business processes for interoperability solution recommendation, which uses the interoperability assessment results from the previous business function.

A Data Consumer starts the interoperability scenario by requesting a specific Data Resource from a Data Provider. The Data Consumer also specifies the expected schema of the Data Resource, which might be incompatible with the current schema specified by the Data Provider. Each schema is stored as Data Self-Description object inside the IDS Connector of each Data Space Participant. If the Data Provider sets the Data Resource as commercial data, the pricing information is also collected by the Interoperability Simulator. These schemas and pricing information are collected as inputs to assess the information gap.

The inputs are further processed by two application functions. The Schema Matching function compares the schemas and creates a mapping from the elements of one schema to the elements of the other schema. Besides comparing schemas between two Data Participants, it is possible to compare a schema with the Standard Data Model stored in Vocabulary Hub. This is useful in an interoperability scenario where a Data Participant needs to convert a Data Resource to a standardized format. The Pricing Calculation function calculates the total costs using the given pricing information. The results from both functions are collected and reported to the Data Consumer, showing the schema mapping results and the total cost to access the Data Resource.

At this point, the Data Consumer can view the interoperability assessment results and needs to find the available solutions for the interoperability scenario. The Interoperability Simulator can use the assessment results to find suitable IDS Apps in the App Store. The assessment results are converted to search criteria that can be used to query the App Store. The query results from the App Store are then forwarded to Data Consumer as solution recommendation results from the Interoperability Simulator.

3.3.2 Implementation and Deployment View

Figure 13 shows the Implementation and Deployment View that describes the relationship between the Application Layer and the Technology Layer. The view shows how the technology infrastructure can realize the application components, without prescribing specific technologies for the technology infrastructure. The Implementation and Deployment View uses another basic ArchiMate viewpoint named the Implementation and Deployment Viewpoint. ArchiMate elements from the Technology Layer that are related to physical objects are not included in the viewpoint, such as the Communication Network element and the Device element (The Open Group, 2022).

The Implementation and Deployment View refers to the same Application Layer components of the Application Usage View. Two additional elements are added to the App Store application component to show the relationship between these App Store's sub-components and the Technology Layer.

As discussed in Section 2.3, IDS Connectors and their sub-components are deployed using containerization. Each sub-component is deployed in a separate container with a separate runtime. The relationship between the IDS Connector and containerization is shown in Figure 13, where the IDS Connector App in the Technology Layer realizes the IDS Connector component in the Application Layer. This relationship also applies to the other IDS components such as Vocabulary Hub and App Store, where each IDS component is realized by its own application component in the Technology Layer. These containers are run in a container runtime environment. Container orchestration is not used to manage the containers, considering the small number of the deployed containers and that we did not consider requirements related to system scalability and reliability in this research.

Data objects in the Application Layer that need a permanent storage are realized by the Relational Database Management System (RDBMS). Data objects that are not realized by RDBMS, such as those that reside in the Interoperability Simulator, are produced and accessed during runtime. Therefore, they do not need permanent storage and are discarded after the Interoperability Simulator finishes its job.

The Interoperability Simulator App performs the schema-matching operation using Matcher Apps chosen from the available Matchers. Each Matcher represents a distinct schema-matching implementation that is deployed in a separate container from the Vocabulary Hub App container. The Interoperability Simulator App can communicate with each Matcher App using a web service that follows a specification that applies to all Matchers, enabling the Interoperability Simulator to have a standard interface to communicate with all Matchers.

Deploying each Matcher in a separate container offers more flexibility to the architecture, since the implementation details of the Matcher can be decoupled from the implementation details of the Interoperability Simulator. This architecture design also considers that every Matcher implementation might be developed by different people with different technology preferences. The only requirements that must be fulfilled by each Matcher are to offer a web service that can be used by the Interoperability Simulator and that the Matcher App is deployed in a container.

Even though the decoupled design of Matchers gives more flexibility to the Matcher implementation and deployment, it also adds more complexity to the deployment phase. The Interoperability Simulator has to make sure that the Matchers are always accessible and give the expected response. Moreover, using a web service as an interface between the Matcher and the Interoperability Simulator adds more latency to the response time of the Interoperability Simulator, compared to using direct procedure calls if the Matcher is implemented as an application module of the Interoperability Simulator. However, the added latency is not expected to give a significant performance degradation to the Interoperability Simulator, while the flexibility offered by the decoupled Matcher design outweighs the inherent drawbacks.

3.3.3 Business Processes

The Business Layer in Figure 12 shows the sequence of business processes of the Interoperability Simulator. However, this sequence only describes high level business processes and their relationships to the other elements in the architecture, and does not describe the specific activities that happen in a business process and the parties involved for each activity. To address this issue, we drew the BPMN diagrams to specify the business processes of the Interoperability Simulator.

The business processes in Figure 12 are grouped into two distinct business functions. The business processes for each business function can be represented in a BPMN diagram. Figure 14 represents the Interoperability Assessment business function, showing a process where a Data Participant (either Data Consumer or Data Provider) needs to perform interoperability assessment using Interoperability Simulator. Figure 15 represents the Interoperability Solution Recommendation business function, describing how the interoperability assessment results from the previous business function can be used to query the IDS App Store. The query results give a list of IDS Apps that can be used to address the specific interoperability scenario.

As mentioned in Section 3.1, an interoperability scenario in IDS can happen during the Data Offering step, when the Data Consumer is searching for a suitable data resource in the data space. However, an interoperability scenario can also happen in another step. IDSA (2022) presents a business process called Use IDS App, where a data participant needs to find the suitable Data Apps to perform data transformation tasks. The Use IDS App business process has been depicted earlier in Figure 8. The Interoperability Simulator can assist the Data Participant to find the Data Apps if the required task is related to data interoperability with the other data participant. Figure 16 illustrates the proposed revision of the Use IDS App business process, where the new business process gives the user a new option to use the Interoperability Simulator to find the relevant IDS Apps. The new business process in Figure 16 incorporates the two business functions in Figure 14 and Figure 15 as subprocesses.

Figure 15 shows the Interoperability Solution Recommendation business function, which relates to the components in white in Figure 12 and Figure 13. Therefore, this function is out of scope of this research and will not be discussed further in this thesis.

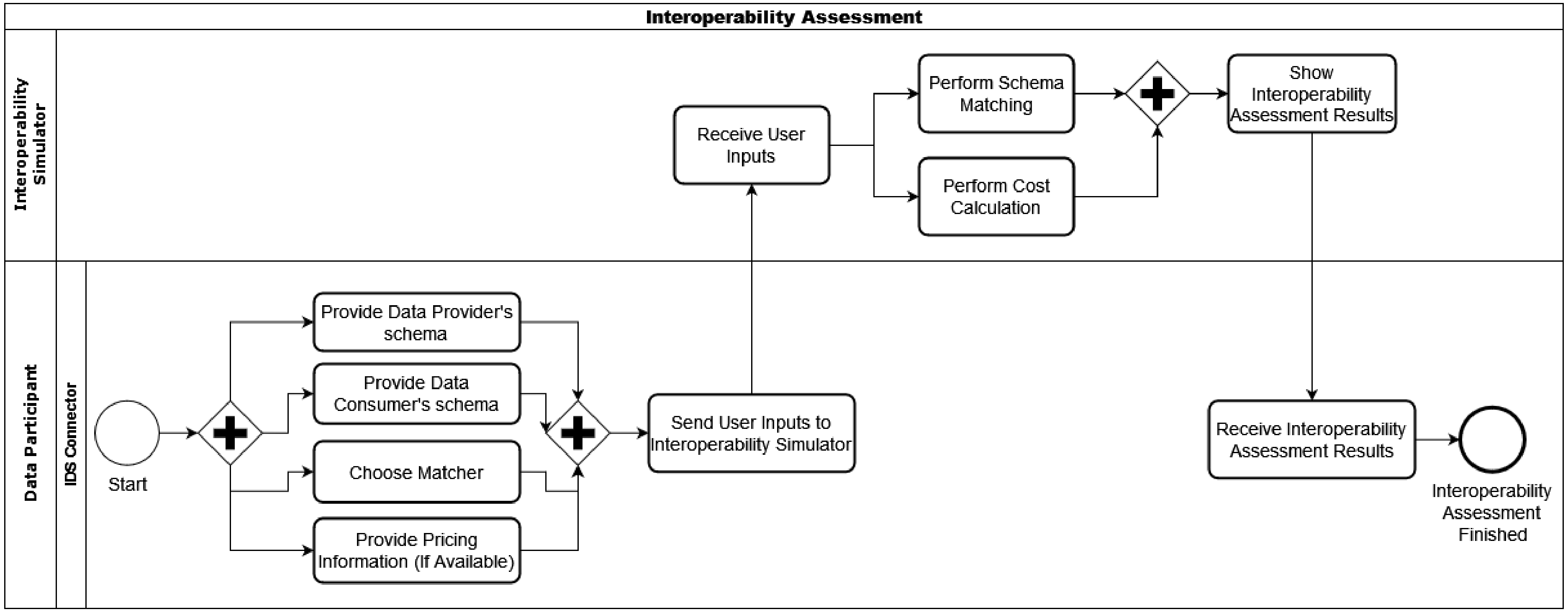

Interoperability Assessment
Interoperability Simulator
Data Participant
IDS Connector
Start
Provide Data Provider's schema
Provide Data Consumer's schema
Choose Matcher
Provide Pricing Information (If Available)
Send User Inputs to Interoperability Simulator
Receive User Inputs
Perform Schema Matching
Perform Cost Calculation
Show Interoperability Assessment Results
Receive Interoperability Assessment Results
Interoperability Assessment Finished

Figure 14 "Interoperability Assessment" business function

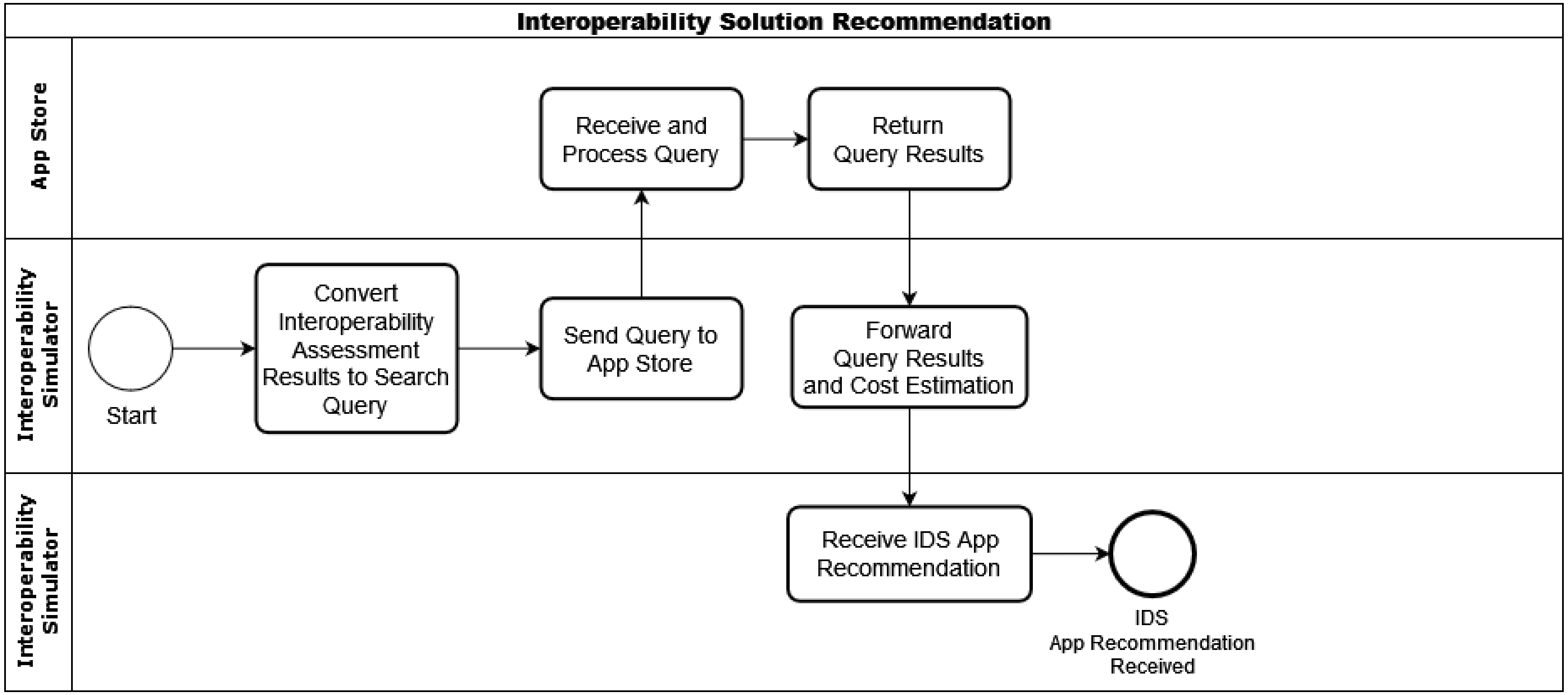

Interoperability Solution Recommendation
App Store
Interoperability Simulator
Interoperability Simulator
Receive and Process Query
Return Query Results
Start
Convert Interoperability Assessment Results to Search Query
Send Query to App Store
Forward Query Results and Cost Estimation
Receive IDS App Recommendation
IDS App Recommendation Received

Figure 15 "Interoperability Solution Recommendation" business function

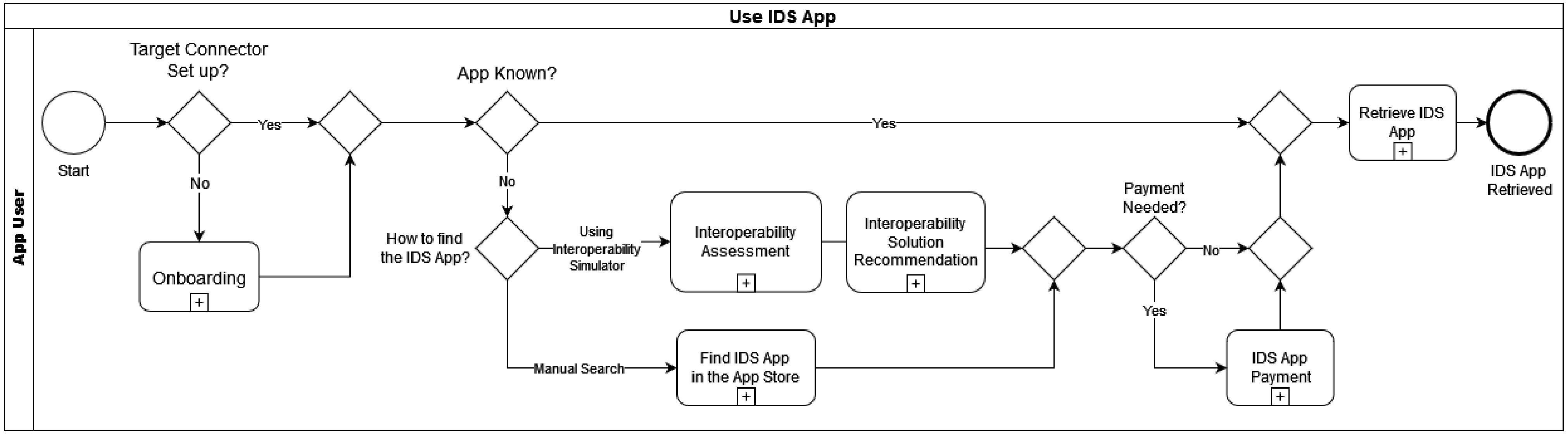


Figure 16 Proposed revision of the "Use IDS App" business process, adding Interoperability Simulator as an alternative way to find an IDS App

4. PROTOTYPE DEVELOPMENT

To demonstrate that the solution design specified in Chapter 3 can solve the identified problems, we perform the Treatment Validation task of the design cycle. This task produces a validation model that consists of a model of the solution and the model of the problem context (Wieringa, 2014). This chapter explains how a prototype that represents a model of the solution was developed.

This chapter begins with Section 4.1 that explains the concrete architecture used to implement the solution. Section 4.2 shows the development of Interoperability Simulator and its Matchers as web services, which are used by the Data Participants to assess an interoperability scenario. Section 4.3 gives details about the integration of the Interoperability Simulator prototype into an existing data space prototype, while Section 4.4 covers the deployment of the Interoperability Simulator using containerization.

4.1 CONCRETE ARCHITECTURE

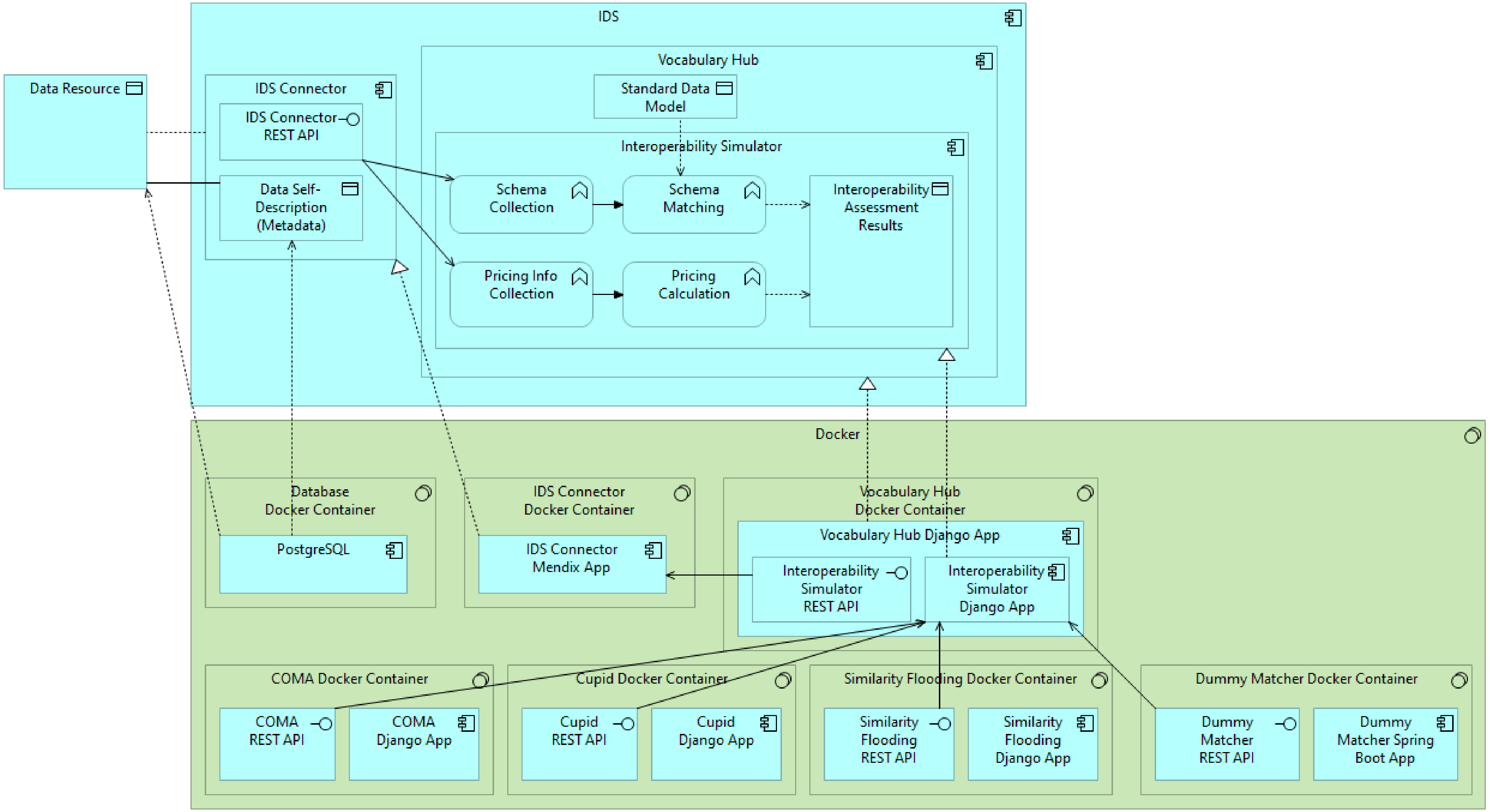
A concrete architecture was developed based on the reference architecture in Section 3.3. The concrete architecture realizes the Implementation and Deployment view of the reference architecture in Figure 13, excluding the components that are related to IDS App Store. Compared to the reference architecture, the Application Layer of the concrete architecture remains unchanged. However, the Technology Layer of the concrete architecture uses a more specific technology stack that replace the general elements specified in the reference architecture. The Implementation and Deployment view of the concrete architecture is illustrated in Figure 17.

The concrete architecture uses Docker as container runtime environment, in which the application components are deployed as Docker containers. The details of the deployment aspect of the prototype are covered further in Section 4.4. Two groups of application components form our prototype:

1. The Interoperability Simulator and the four Matchers, which form the core part of the interoperability assessment functionality. Each component is deployed either as a Python Django web application or as a Java Spring Boot web application. These web applications communicate with each other using REST APIs. Further details about the Interoperability Simulator and the Matchers are given in Section 4.2.

2. The IDS Connectors and their databases, which provide the data Resources and the user interface for the Interoperability Simulator. The user interface of the IDS Connectors (both the Data Provider and the Data Consumer) are implemented using Mendix[1], with PostgreSQL as Database Management System. The IDS Connectors communicate with the Interoperability Simulator using a REST API. Section 4.3 discusses the implementation of the IDS Connectors and the databases in more depth.

[1] https://www.mendix.com/


IDS
Data Resource
IDS Connector
IDS Connector REST API
Data Self-Description (Metadata)
Vocabulary Hub
Standard Data Model
Interoperability Simulator
Schema Collection
Schema Matching
Interoperability Assessment Results
Pricing Info Collection
Pricing Calculation
Docker
Database Docker Container
PostgreSQL
IDS Connector Docker Container
IDS Connector Mendix App
Vocabulary Hub Docker Container
Vocabulary Hub Django App
Interoperability Simulator REST API
Interoperability Simulator Django App
COMA Docker Container
COMA REST API
COMA Django App
Cupid Docker Container
Cupid REST API
Cupid Django App
Similarity Flooding Docker Container
Similarity Flooding REST API
Similarity Flooding Django App
Dummy Matcher Docker Container
Dummy Matcher REST API
Dummy Matcher Spring Boot App
Figure 17 Implementation and Deployment View of the concrete architecture

4.2 IMPLEMENTATION OF INTEROPERABILITY SIMULATOR

As a standalone application component, the Interoperability Simulator has been developed as a web application that receives user input to solve an interoperability scenario and gives the assessment results back to the user. According to the Application Function of the concrete architecture, the Interoperability Simulator has two distinct functions, which are the schema matching function and the pricing calculation function. These main functionalities are further explained in the sequel.

4.2.1 Schema Matching Function

The purpose of the schema matching function is to perform schema matching operations based on user input. To perform the schema matching operations, the Interoperability Simulator utilizes several schema-matching approaches called Matchers. For each schema matching operation, the user supplies two input strings that represent the source schema and the target schema. These two schemas are forwarded to each Matcher, which return a mapping of elements from the source schema to the target schema. The mapping is represented by a list of pairs of source element and target element, along with the similarity score. The Interoperability Simulator aggregates the mapping from each Matcher to be returned as output. The representation of the mapping is similar to the schema matching depicted in Figure 9.

Each Matcher provides a REST API that can be used by the Interoperability Simulator. Each Matcher conforms to the same API contract, which defines things such as the available API endpoints, the expected structure of the request body, and the possible responses. Currently one API endpoint needs to be implemented by all Matchers and its specification is shown in Table 3. This API endpoint is used exclusively by the Interoperability Simulator and cannot be used directly by the Data Consumer.

Table 3 Matcher API contract

Endpoint	**`POST /matcher/match-schemas`**
Description	Given a source schema and a target schema, returns a schema mapping along with the similarity score.
Request Headers	`Content-Type: application/json`

Request Body	**`source_schema`**: String. Required. Comma Separated Values (CSV) header that contains the source schema elements.
	`target_schema`: String. Required. Comma Separated Values (CSV) header that contains the target schema elements.
	
	Example:
	`{`
	`  "source_schema": "EID,Writers,Cited by,Title,Year,zipcode",`
	`  "target_schema": "EID,cited-by,Schrijvers,Country,postcode"`
	`}`

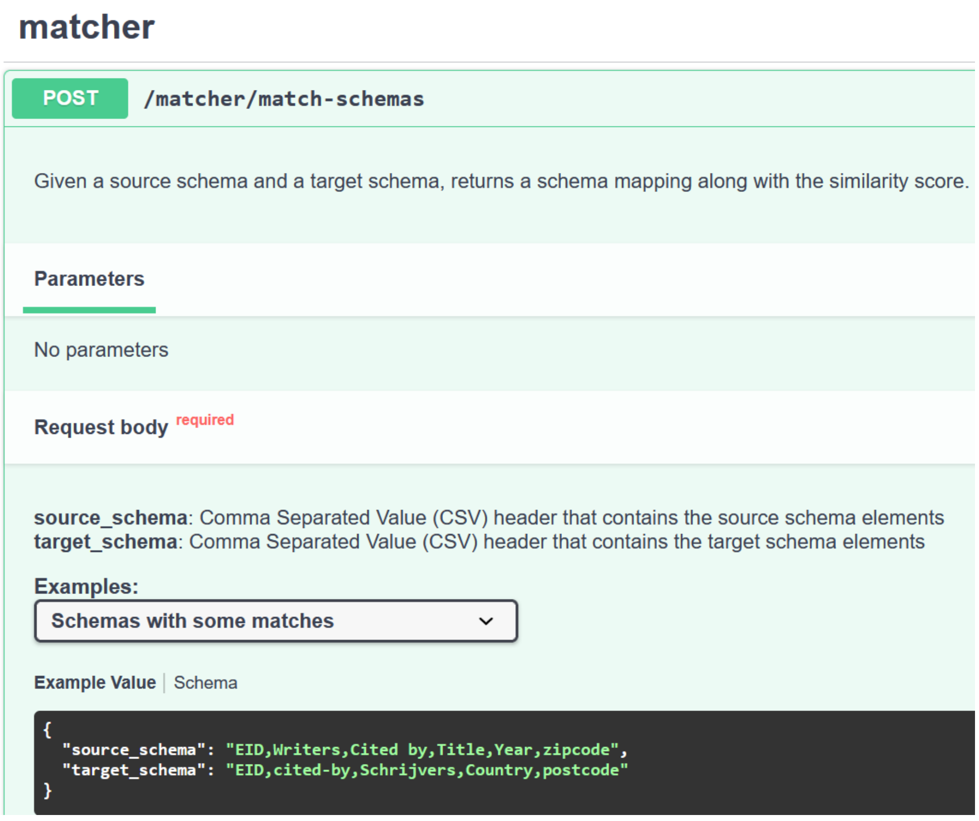
Response	Array of JSON objects that consists of these attributes: **`source_element`**: String. The source element that is mapped to `target_element`. **`target_element`**: String. The target element that is mapped from `source_element`. **`score:`** Decimal String. The similarity score of the element mapping, ranging from 0 to 1. Example:

```
[
  {
    "source_element": "EID",
    "target_element": "EID",
    "score": "0.807"
  },
  {
    "source_element": "Cited by",
    "target_element": "cited-by",
    "score": "0.645"
  }
]
```

The API contract of a Matcher has been written using OpenAPI version 3.0.3[2] and is stored as a JSON file. The schema file is rendered using Swagger UI[3] and is accessible using the following URL:

`http://matcher-host:port/docs`

where `matcher-host` and `port` represent the host address and the port number of the Matcher, respectively. Figure 18 shows the example of Matcher API documentation rendered using Swagger UI.


matcher

POST /matcher/match-schemas

Given a source schema and a target schema, returns a schema mapping along with the similarity score.

Parameters

No parameters

Request body required

source_schema: Comma Separated Value (CSV) header that contains the source schema elements
target_schema: Comma Separated Value (CSV) header that contains the target schema elements

Examples:
Schemas with some matches

Example Value | Schema

```
{
  "source_schema": "EID,Writers,Cited by,Title,Year,zipcode",
  "target_schema": "EID,cited-by,Schrijvers,Country,postcode"
}
```

Figure 18 OpenAPI documentation of a Matcher rendered using Swagger UI

Using the API contract, a new Matcher can be developed and integrated into the Interoperability Simulator. A Matcher can be developed using any programming language or framework as long as it conforms to the REST API contract. A Dummy Matcher has been

[2] https://spec.openapis.org/oas/v3.0.3
[3] https://swagger.io/tools/swagger-ui/

implemented for demonstration purposes using the Spring Boot framework[4]. It conforms to the API contract by accepting the endpoint and the request body as specified in the contract. However, it does not perform any meaningful schema matching operation since it always gives an empty array as the output.

Besides the Dummy Matcher, we used an existing Matcher implementation called Valentine, which was developed by TU Delft Data Management Group[5]. The Matcher is implemented as a Python package and is based on research that aims to evaluate schema matching techniques using tabular data (Koutras et al., 2021). Valentine has the three Matcher groups shown in Table 4.

Table 4 Three Matcher groups in Valentine implementation

Matcher Group	Matcher Name
Schema-based Matchers	Cupid, Similarity Flooding
Instance-based Matchers	Distribution-based, Jaccard-Levenshtein
Schema-based and Instance-based Matcher	COMA

As explained in Section 3.1, an interoperability scenario in IDS can only be solved using schema-level Matchers. Therefore, only COMA, Cupid, and Similarity Flooding from the Valentine implementation that are relevant to our research. Section 2.6.4 explains how each of these three Matchers works. In the Valentine project repository, each Matcher is implemented as a separate Python module, making the codebase easier to extract. To enable CSV processing using the Valentine Matchers, the extracted codebase still needs a few adjustments, as Valentine processes the inputs as `pandas DataFrame`[6]. For example, two new Python classes are implemented to serialize data from user input to the Matcher's internal representation. The two classes replace the existing serialization implementation that utilize `pandas DataFrame` instead of CSV. These adjustments remove the dependency of the `pandas` package, which is not used anywhere else in the Matcher.

After the Matcher codebase has been extracted from Valentine, the next step is to create an interface to enable communication between the Matcher and the Interoperability Simulator. Since the three Matchers are already implemented in Python, the Django REST framework[7] was chosen as REST API application framework. Django was chosen over other Python frameworks such as Flask due to our familiarity with the framework. Each Matcher is wrapped as a Django application, using the API contract as a guide to implement the API. Together with the Dummy Matcher, four Matchers can therefore be used by the Interoperability Simulator as web services.

The following steps summarize the actions taken to extract a Matcher from Valentine:

1. Copy the Python module of the Matcher along with its dependencies such as Valentine serialization classes.

2. Modify the Matcher dependencies to enable data serialization using CSV instead of `pandas DataFrame`.

[4] https://spring.io/projects/spring-boot
[5] https://github.com/delftdata/valentine
[6] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
[7] https://www.django-rest-framework.org/

3. Put the Matcher Python module into an empty Django app, then use the module to perform schema matching based on the input values of the API requests.

4.2.2 Pricing Calculation Function

A Data Consumer that wants to access a commercial Data Resource needs to be aware of the price incurred by the Data Provider. As mentioned in Section 3.3.1, the Interoperability Simulator needs to collect this pricing information and calculate the total cost so that the Data Consumer can be informed with the monetary cost of accessing the Data Resource. The pricing calculation function is performed by the Interoperability Simulator, which receives pricing information as input and uses it to calculate the total cost.

The IDS Reference Architecture Model does not specify the details about this pricing information. Therefore, this research proposes two simple payment models as starting points:

1. **One-time payment model**, in which the Data Consumer performs a single payment to the Data Provider. For example, a Data Consumer needs to pay €500 upfront before being granted access to a dataset of shipping companies.

2. **Recurring payment model**, in which the Data Consumer performs a repeated payment over a set interval such as months or years. For example, a Data Consumer is required to pay €50 per month over two years (24 months) to gain access to transport consignment data from the Data Provider's company, resulting in a total cost of €1200 to access the data for two years.

We assume that each payment model does not consider additional parameters such as taxes and discounts. Currently the pricing calculation function is not implemented as a separate application component due to its simple calculation logic, so it is coupled with the main logic of the Interoperability Simulator. However, it is possible to create a dedicated application component for the pricing calculation function when more realistic and sophisticated payment models are incorporated to the IDS environment.

4.2.3 Combining the Schema Matching and the Pricing Calculation Functions

After the schema matching function and the pricing calculation function are finished with their respective processes, their results are aggregated by the Interoperability Simulator. For each Matcher, the Interoperability Simulator creates two groups of schema elements, namely the matched elements that have been identified by the Matcher, and the unmatched elements that do not have any potential matches according to the Matcher. The relationship between the matched elements and the unmatched elements can be expressed using the following formulas

$$elements = matched \cup unmatched$$

$$matched \cap unmatched = \emptyset$$

where $elements$ refers to the set of all the schema elements, $matched$ refers to the set of the matched elements, and $unmatched$ refers to the set of the unmatched elements. The matched elements and the unmatched elements are mutually exclusive, therefore one schema element cannot be identified as both a matched element and an unmatched element. The above formulas apply to both the elements of the source schema and the target schema.

The entire process of combining the schema matching function and the pricing calculation function is performed when the Data Consumer calls an API endpoint to perform an interoperability assessment. This API endpoint provided by the Interoperability Simulator is described in Table 5.

Table 5 API contract of the Interoperability Simulator

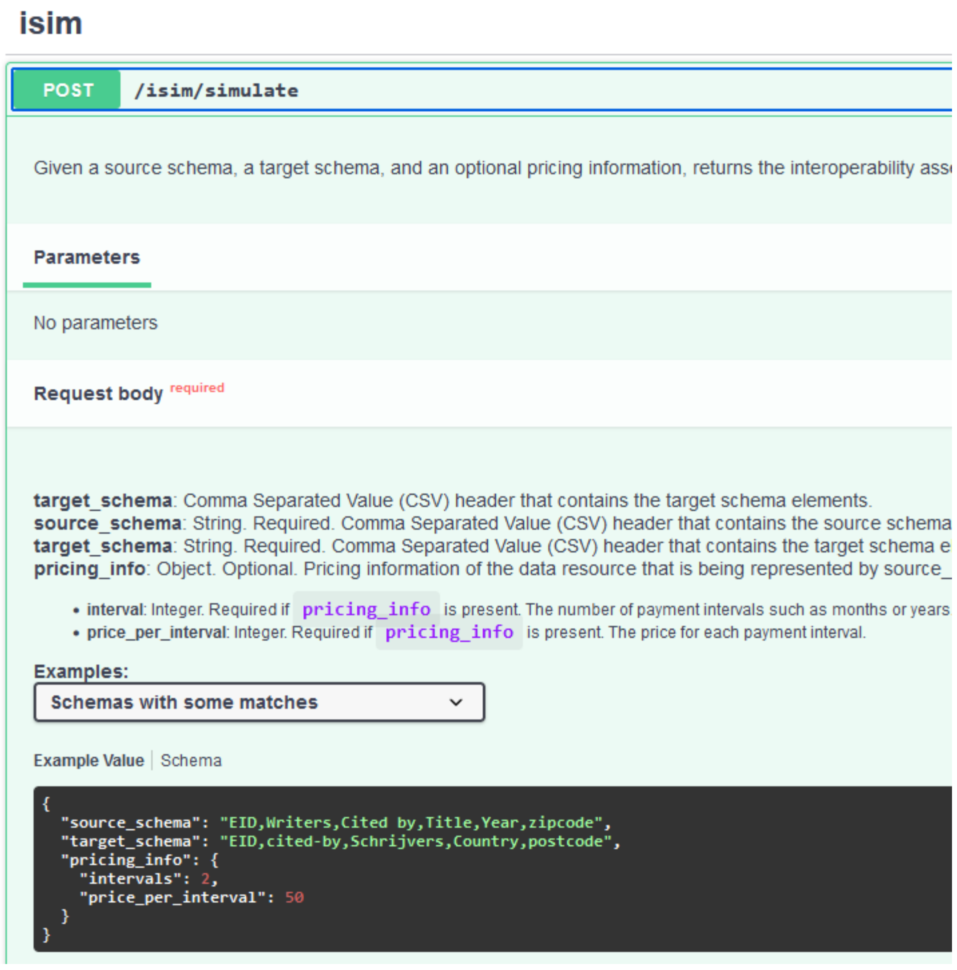
Endpoint	**`POST /isim/simulate`**
Description	Given a source schema, a target schema, and optional pricing information, returns interoperability assessment results that consist of schema mapping results and total price of the data resource.
Request Headers	`Content-Type: application/json`
Request Body	**`source_schema`**: String. Required. Comma Separated Values (CSV) header that contains the source schema elements. **`target_schema`**: String. Required. Comma Separated Values (CSV) header that contains the target schema elements. **`pricing_info`**: Object. Optional. Pricing information of the data resource represented by `source_schema`. The object consists of the following attributes: **`interval`**: Integer. Required if **`pricing_info`** is present. The number of payment intervals, such as months or years. **`price_per_interval`**: Integer. Required if **`pricing_info`** is present. The price for each payment interval. Example: `{` `"source_schema": "EID,Writers,Cited by,Title,Year,zipcode",` `"target_schema": "EID,cited-by,Schrijvers,Country,postcode",` `"pricing_info": {` `"intervals": 2,` `"price_per_interval": 50` `}` `}`
Response	A JSON object that consists of these attributes: **`matcher_results`**: Array of JSON objects. Each item consists of: **`matcher`**: String. The name of the Matcher. **`matched_elements`**: Array of JSON objects. Each item consists of: **`source_element`**: String. The source element of **`source_schema`** that is mapped to `target_element`. **`target_element`**: String. The target element of **`target_schema`** that is mapped from `source_element`. **`score:`** Decimal String. The similarity score of the element mapping, ranging from 0 to 1. **`unmatched_elements`**: JSON object that consists of: **`source_elements`**: String. The unmatched elements of **`source_schema`**, separated by comma. **`target_elements`**: String. The unmatched elements of **`target_schema`**, separated by comma. Example: `{` `"matcher_results": [` `{` `"matcher": "coma",` `"matched_elements": [` `{` `"source_element": "EID",` `"target_element": "EID",`

```
          "score": "0.807"
        },
        {
          "source_element": "Cited by",
          "target_element": "cited-by",
          "score": "0.645"
        },
      ],
      "unmatched_elements": {
        "source_elements": "Title,Year",
        "target_elements": "Country"
      }
    }
  ],
  "total_price": 100
}
```

The API contract of the Interoperability Simulator is also specified using OpenAPI version 3.0.3. The schema file is also rendered using Swagger UI and is accessible through the following URL:

`http://isim-host:port/docs`

where `isim-host` and `port` represent the host address and the port number of the Interoperability Simulator application, respectively. Figure 19 shows the Interoperability Simulator API documentation rendered using Swagger UI.


isim

POST /isim/simulate

Given a source schema, a target schema, and an optional pricing information, returns the interoperability ass

Parameters

No parameters

Request body required

target_schema: Comma Separated Value (CSV) header that contains the target schema elements.
source_schema: String. Required. Comma Separated Value (CSV) header that contains the source schema
target_schema: String. Required. Comma Separated Value (CSV) header that contains the target schema e
pricing_info: Object. Optional. Pricing information of the data resource that is being represented by source_

- **interval**: Integer. Required if `pricing_info` is present. The number of payment intervals such as months or years
- **price_per_interval**: Integer. Required if `pricing_info` is present. The price for each payment interval.

Examples:
Schemas with some matches

Example Value | Schema

```
{
  "source_schema": "EID,Writers,Cited by,Title,Year,zipcode",
  "target_schema": "EID,cited-by,Schrijvers,Country,postcode",
  "pricing_info": {
    "intervals": 2,
    "price_per_interval": 50
  }
}
```

Figure 19 OpenAPI documentation of the Interoperability Simulator rendered using Swagger UI

Figure 20 presents the interactions that happen when a Client send an interoperability assessment request to the Interoperability Simulator. The term *Client* is used because the actual entity that initiates the request might be a human user or a software component (such as a front-end web application). The interaction starts when the Client consumes the API endpoint of the Interoperability Simulator by sending an HTTP POST request, along with the required request headers and request body as specified in Table 5. If pricing information is present in the request body, the Interoperability Simulator calculates the total price of the data resource based on the pricing information. The pricing calculation step is optional, as indicated by the *Opt* fragment in the sequence diagram.

Regardless of the presence of pricing information, the Interoperability Simulator proceeds to interact with the available Matchers. For each Matcher, the Interoperability Simulator requests a schema matching operation by sending an HTTP POST request to the Matcher, as specified before in Table 3. After receiving a response from the Matcher, the Interoperability Simulator processess the response further by grouping the matched and the unmatched schema elements. This process is repeated for each available Matcher, resulting in a loop as indicated by the *Loop* fragment in the sequence diagram.

The processed responses from all Matchers are then aggregated into a single response, which is combined with the pricing information (if available). Finally, the combined result is returned to the Client as an Interoperability Simulator API response.

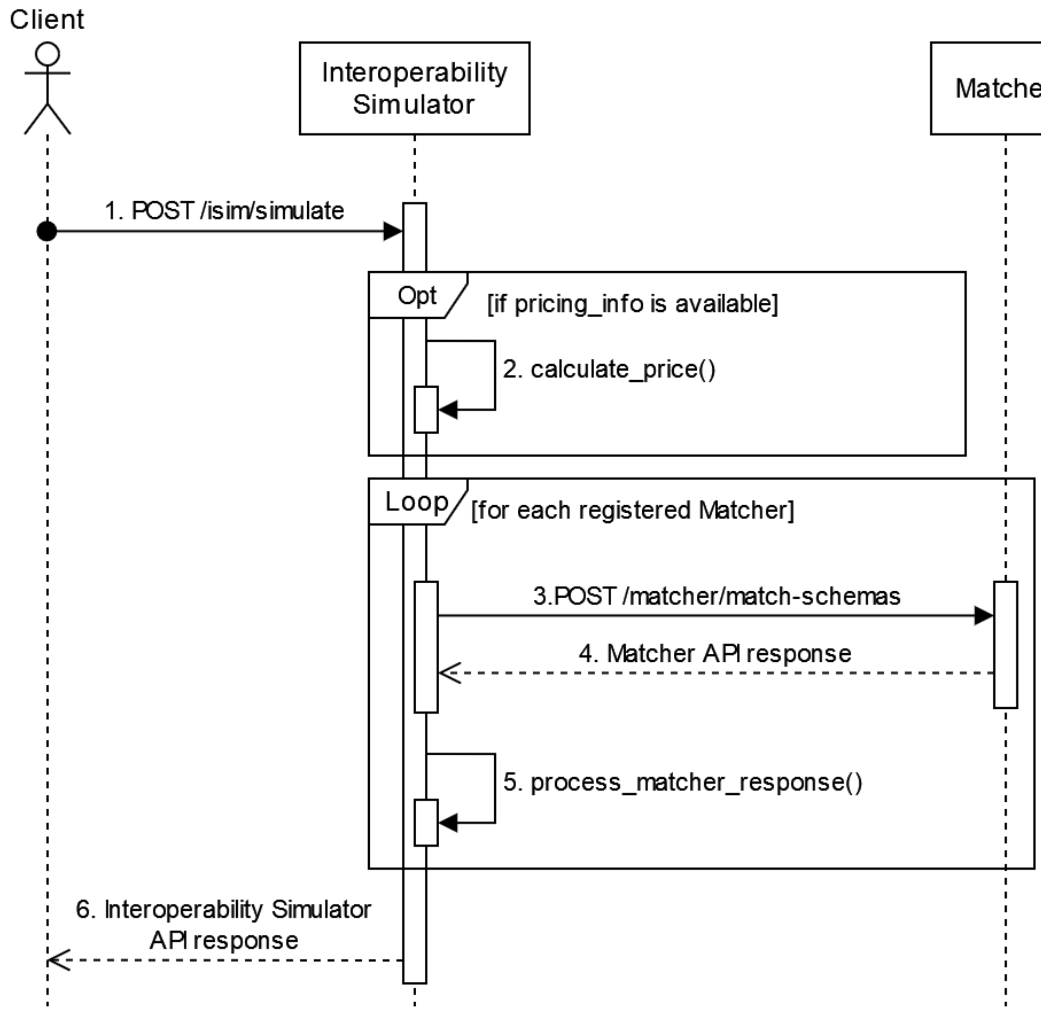

Client
Interoperability Simulator
Matcher
1. POST /isim/simulate
Opt
[if pricing_info is available]
2. calculate_price()
Loop
[for each registered Matcher]
3.POST /matcher/match-schemas
4. Matcher API response
5. process_matcher_response()
6. Interoperability Simulator API response

Figure 20 Sequence diagram of the interactions between Client, Interoperability Simulator, and Matchers

The current prototype implementation uses a synchronous approach to call the API endpoints of the Matcher. This approach is straightforward and can ensure the order of the API responses. However, the synchronous approach blocks the Interoperability Simulator after each API call until a response returns, preventing the Interoperability Simulator from doing other tasks in the meantime. This means that, other API calls can only be executed after the response from the blocked API call has been received. Since in this research we ignored the scalability aspect of the solution, synchronous approach is acceptable.

4.3 IMPLEMENTATION OF THE IDS CONNECTOR

The reference architecture of Figure 13 prescribes a few application components that form an IDS data space. The IDS Connector is an application component that serves as an interface between data space participants who want to interact with each other. The IDS Reference Architecture Model does not provide a working prototype of the IDS Connector, so in this research we reused an existing IDS prototype implementation that was developed for an Engineering Doctorate project at the University of Twente (Firdausy, 2023).

The IDS prototype comprises several components, such as IDS Connector, IDS Data Apps, and a new component called Connector Store that is not an official IDS component. In this research, we only reuse the IDS Connector implementation, as IDS Data Apps and Connector Store are out of scope. Since the IDS Connector prototype has been already implemented using Mendix, it would be more efficient to extend the existing Mendix implementation to support the functionalities of the Interoperability Simulator. The decision of using PostgreSQL as the RDBMS of the IDS Connector is also influenced by the existing IDS Connector implementation.

The different functionalities that are related to the Interoperability Simulator are further discussed in the sequel.

4.3.1 IDS Connector Data Model

Mendix uses a data model called Domain Model to describe the entities in a Mendix application. This model is similar to the Entity Relationship Diagram (ERD) that describes the tables in a database, along with the constraints and relationships with other tables.

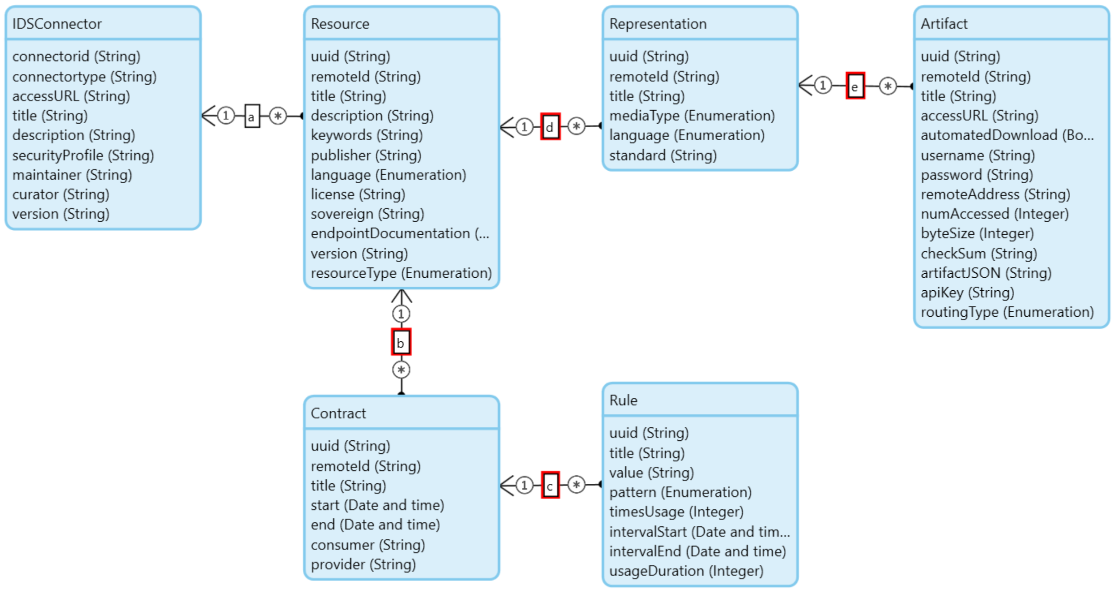
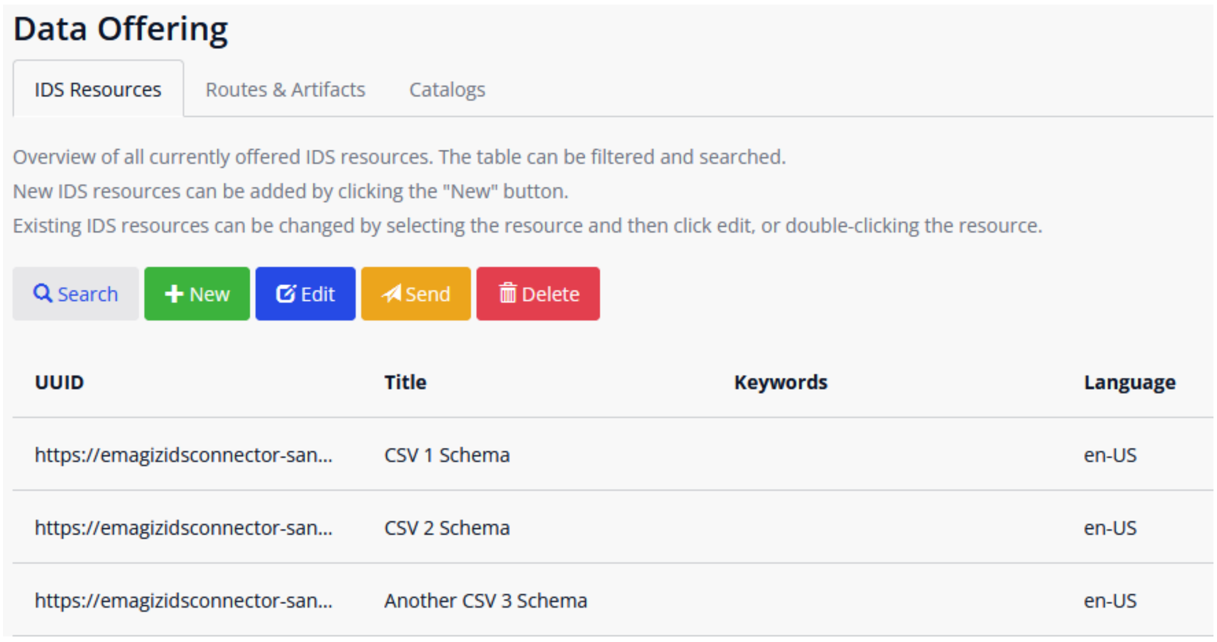


Figure 21 Domain Model diagram of an IDS Connector and its related entities

Figure 21 presents a Mendix Domain Model diagram of the IDS Connector according to the prototype described in Firdausy (2023). The diagram only covers the most relevant entities such as `IDSConnector`, `Resource`, and `Artifact`. An arrow between two entities represents a relationship. For example, an arrow labelled `a` represents a one-to-many relationship between `IDSConnector` and `Resource`. In the sequel we explain how the Domain Model is used to realize some functionalities of an IDS Connector.

4.3.2 IDS Connector Data Offering by Data Provider

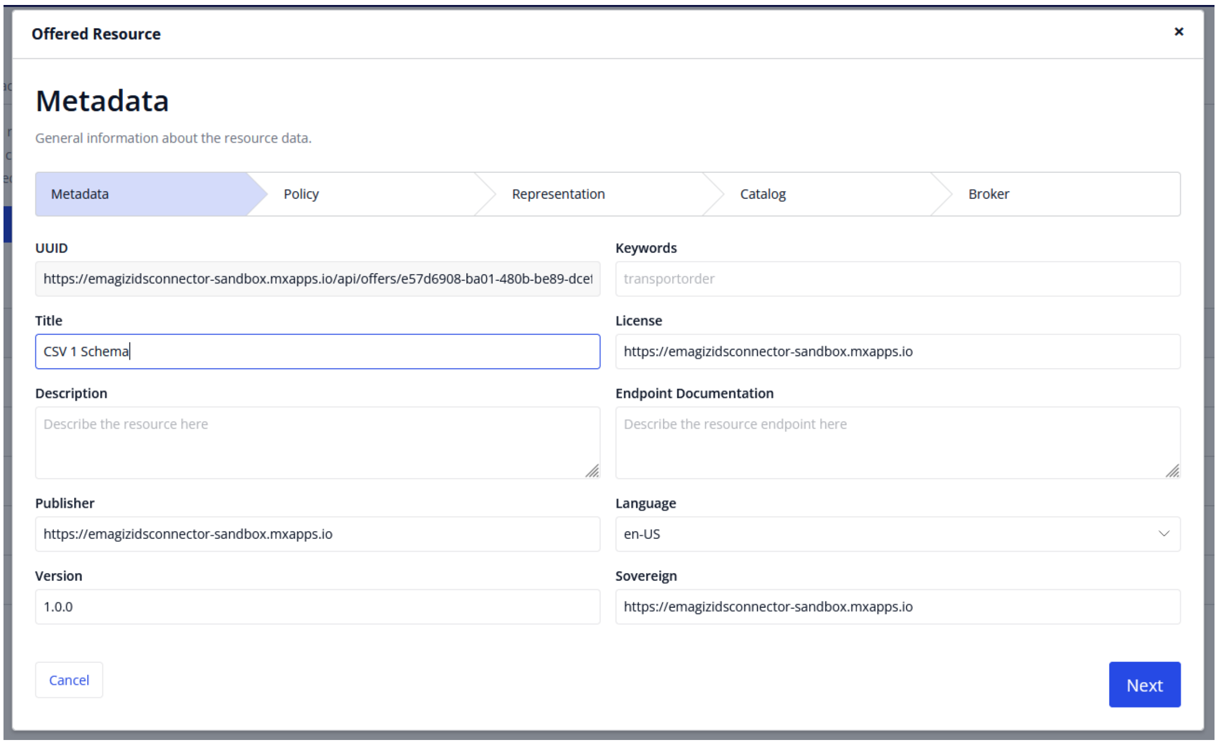
In a data space, a Data Provider can publish data resources through the IDS Connector by using the Data Offering functionality. Figure 22 shows the Data Offering Overview page that lists all the resources offered by the Data Provider. The Data Provider can perform content management tasks such as creating a new resource, editing an existing resource, or deleting a specific resource. When editing a resource, a pop-up page that shows the metadata of the resource is shown to the Data Provider, as displayed in Figure 23. The metadata contains information such as the UUID, title, description, and version number of the offered resource. These metadata attributes are based on the specification of the `Resource` entity, as depicted in the domain model diagram in Figure 21.



Overview of all currently offered IDS resources. The table can be filtered and searched.
New IDS resources can be added by clicking the "New" button.
Existing IDS resources can be changed by selecting the resource and then click edit, or double-clicking the resource.

UUID	Title	Keywords	Language
https://emagizidsconnector-san...	CSV 1 Schema		en-US
https://emagizidsconnector-san...	CSV 2 Schema		en-US
https://emagizidsconnector-san...	Another CSV 3 Schema		en-US

Figure 22 Data Offering Overview page of the IDS Connector


Offered Resource

Metadata

General information about the resource data.

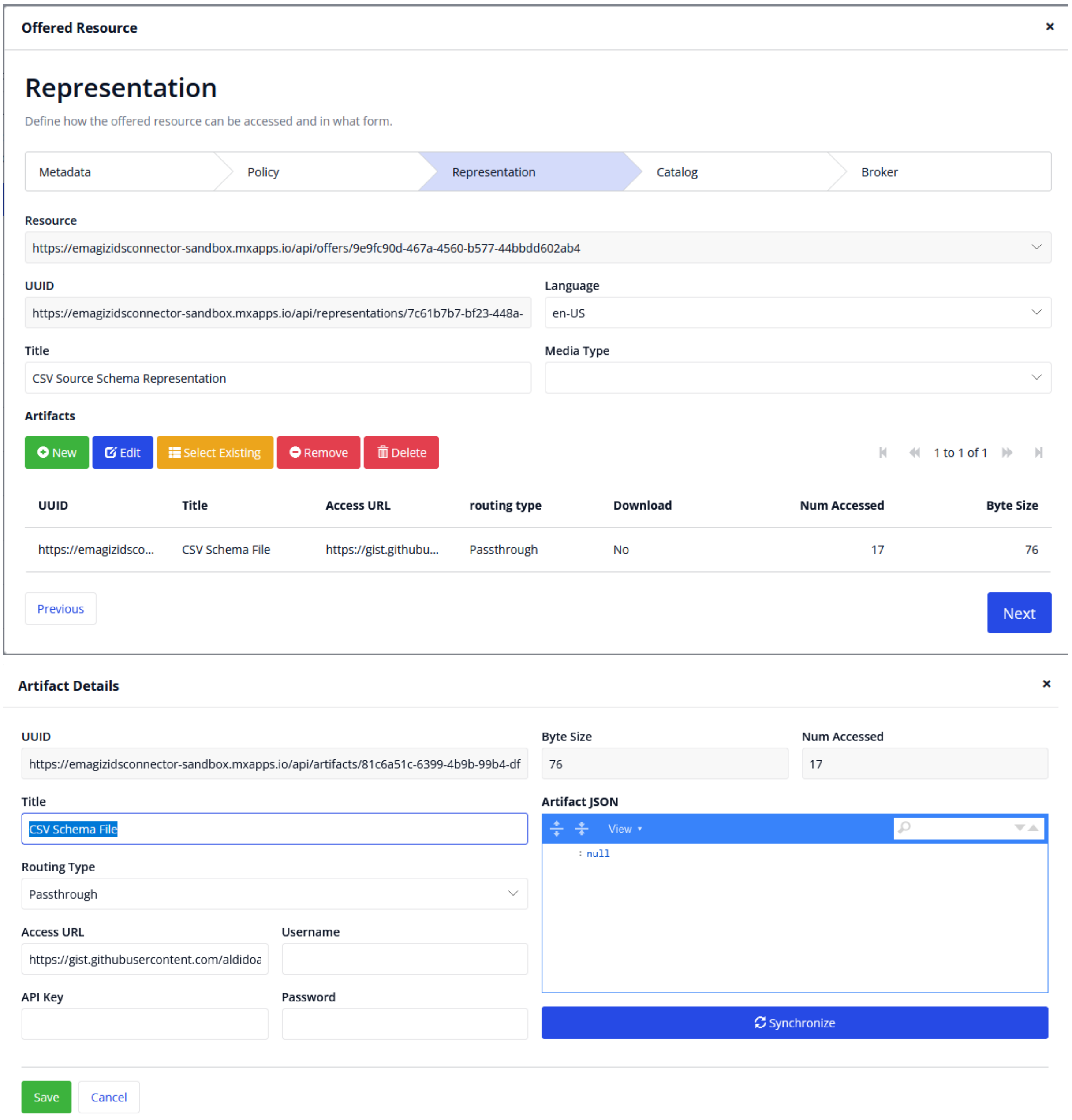
UUID	https://emagizidsconnector-sandbox.mxapps.io/api/offers/e57d6908-ba01-480b-be89-dce
Keywords	transportorder
Title	CSV 1 Schema
License	https://emagizidsconnector-sandbox.mxapps.io
Description	Describe the resource here
Endpoint Documentation	Describe the resource endpoint here
Publisher	https://emagizidsconnector-sandbox.mxapps.io
Language	en-US
Version	1.0.0
Sovereign	https://emagizidsconnector-sandbox.mxapps.io

Figure 23 Pop-up page that displays the metadata of an offered resource

Besides the metadata, an offered resource might also have data usage policies attached to it, which are set by the Data Provider, and define how the offered resource can be used by the Data Consumer. For example, the Data Provider can either grant an unlimited access to the resource or set an expiration date. To simplify the prototype implementation, it is assumed that the Data Provider always provides unlimited access to the offered schemas. In the domain model of Figure 21, a data usage policy is represented by the `Contract` and `Rule` entities.

Besides general metadata and usage policies, an offered resource can also have one or more Representations. A Representation describes another set of metadata, such as media type (e.g., `application/json` or `application/xml`), language, and standard descriptions (e.g., Open Trip Model (OTM) or GS1) (Firdausy, 2023). If an offered resource has multiple Representations, this means that the resource is being offered in multiple forms. For example, a data schema about logistics transportation can be offered in either Dutch or English using JSON or XML format.

Each Representation comprises one or multiple Artifacts that contain the metadata to access the actual offered resource. For example, an Artifact metadata can contain the access URL of the data, along with authentication credentials such as API Key, username, and password. The prototype implementation assumes that an offered resource always has exactly one Representation, which has exactly one Artifact. Figure 24 shows two pop-up pages that display the metadata of a Representation and an Artifact.


Offered Resource

Representation

Define how the offered resource can be accessed and in what form.

Metadata | Policy | Representation | Catalog | Broker

Resource: https://emagizidsconnector-sandbox.mxapps.io/api/offers/9e9fc90d-467a-4560-b577-44bbdd602ab4

UUID: https://emagizidsconnector-sandbox.mxapps.io/api/representations/7c61b7b7-bf23-448a-

Language: en-US

Title: CSV Source Schema Representation

Media Type:

Artifacts

1 to 1 of 1

UUID	Title	Access URL	routing type	Download	Num Accessed	Byte Size
https://emagizidsco...	CSV Schema File	https://gist.githubu...	Passthrough	No	17	76

Artifact Details

UUID: https://emagizidsconnector-sandbox.mxapps.io/api/artifacts/81c6a51c-6399-4b9b-99b4-df

Byte Size: 76

Num Accessed: 17

Title: CSV Schema File

Artifact JSON: : null

Routing Type: Passthrough

Access URL: https://gist.githubusercontent.com/aldidoa

Username:

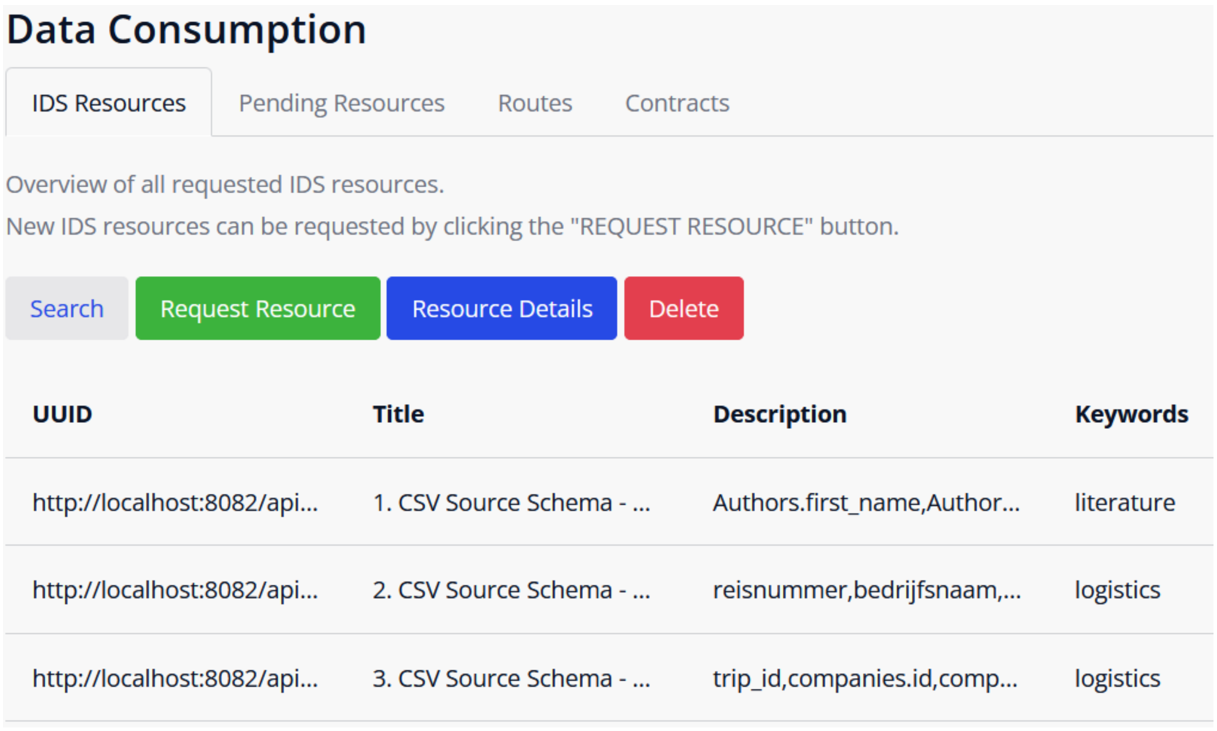
API Key:

Password:

Figure 24 Pop-up pages that display the metadata of a Representation and an Artifact, respectively

4.3.3 IDS Connector Data Consumption by Data Consumer

Once the offered resources have been published by the Data Provider, the Data Consumer can access the resources by issuing a data consumption request. Figure 25 shows the Data Consumption Overview page that lists all the consumed resources by the Data Consumer. The Data Consumer can use this page to request a new resource or to view the details of the existing consumed resources.

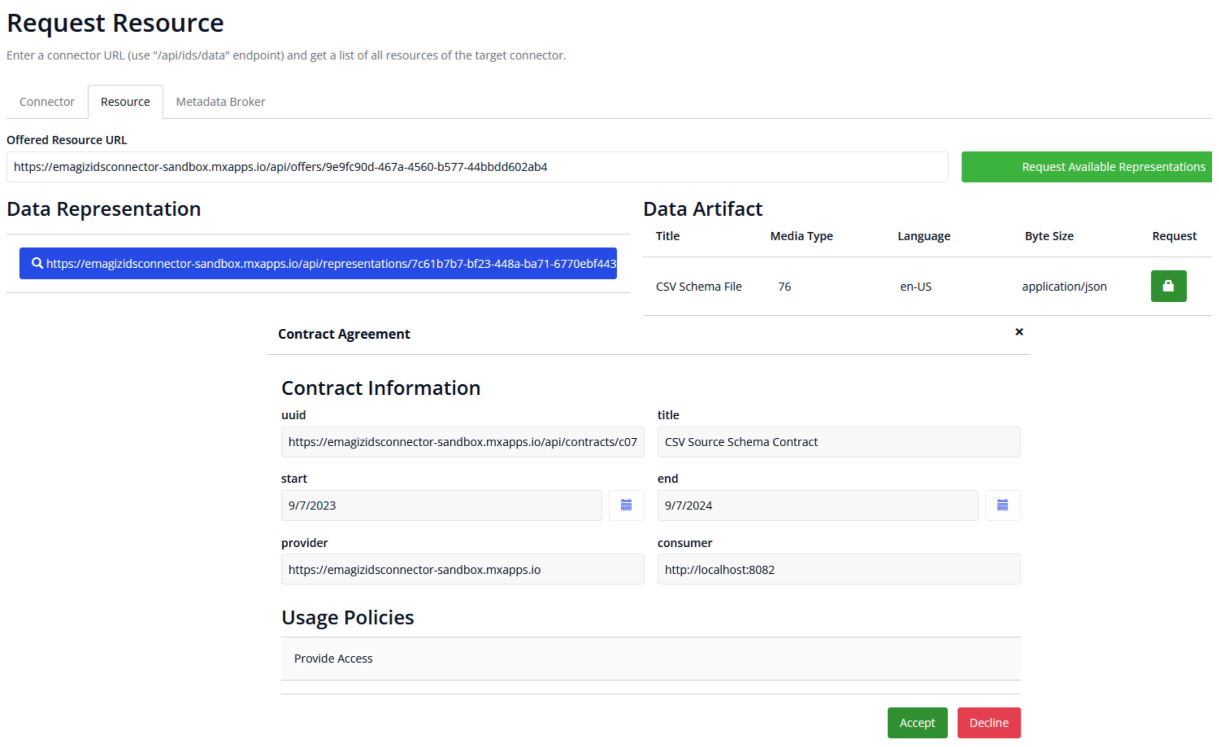


Overview of all requested IDS resources.
New IDS resources can be requested by clicking the "REQUEST RESOURCE" button.

UUID	Title	Description	Keywords
http://localhost:8082/api...	1. CSV Source Schema - ...	Authors.first_name,Author...	literature
http://localhost:8082/api...	2. CSV Source Schema - ...	reisnummer,bedrijfsnaam,...	logistics
http://localhost:8082/api...	3. CSV Source Schema - ...	trip_id,companies.id,comp...	logistics

Figure 25 Data Consumption Overview page of the IDS Connector

The prototype implementation of Firdausy (2023) allows a Data Consumer to request an offered resource using three different ways, namely by using the IDS Connector URL, using the Resource URL, or by using a Metadata Broker. It is assumed that the Data Consumer already knows the URLs of the offered resources. Therefore, the Metadata Broker is not necessary and the Data Consumer can request an offered resource directly using the Resource URL. Figure 26 shows the user interface of the data consumption process using the Resource URL, in which the Data Consumer is presented with the information about the Representation and the usage policy of the requested resource. After the Data Consumer accepts the usage policies of the resource, the Data Consumer gains access to the resource and the resource is added to the Data Consumption Overview page.


Request Resource

Enter a connector URL (use "/api/ids/data" endpoint) and get a list of all resources of the target connector.

Offered Resource URL: https://emagizidsconnector-sandbox.mxapps.io/api/offers/9e9fc90d-467a-4560-b577-44bbdd602ab4

Data Representation

https://emagizidsconnector-sandbox.mxapps.io/api/representations/7c61b7b7-bf23-448a-ba71-6770ebf443

Data Artifact

Title	Media Type	Language	Byte Size	Request
CSV Schema File	76	en-US	application/json	

Contract Agreement

Contract Information

uuid: https://emagizidsconnector-sandbox.mxapps.io/api/contracts/c07	title: CSV Source Schema Contract
start: 9/7/2023	end: 9/7/2024
provider: https://emagizidsconnector-sandbox.mxapps.io	consumer: http://localhost:8082

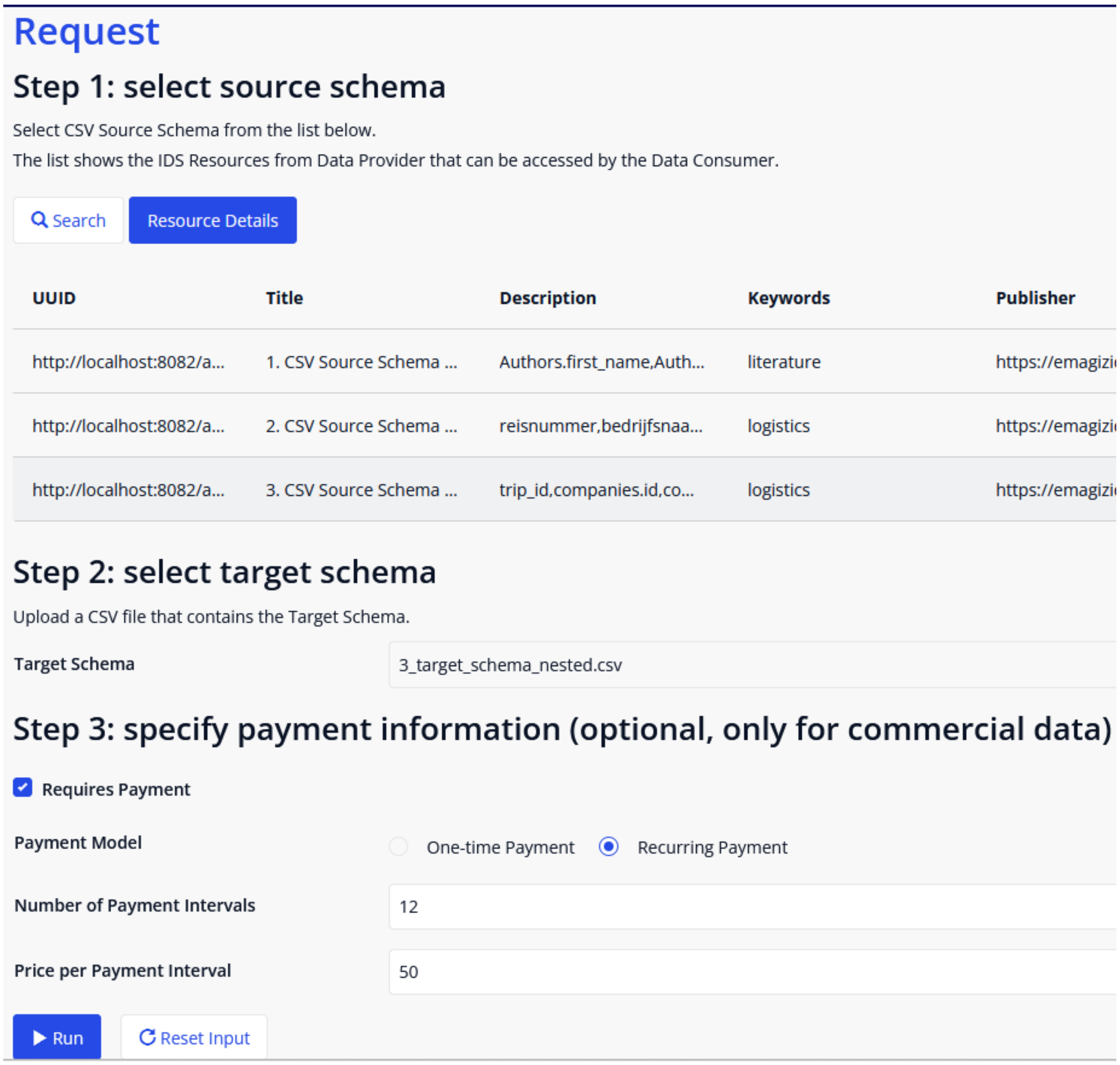
Usage Policies

Provide Access

Figure 26 Pop-up pages that show the data consumption process using Resource URL

4.3.4 IDS Connector Interoperability Assessment for Data Consumer

After the Data Consumer has gained access to the data resource offered by the Data Provider, the Data Consumer may wish to assess the interoperability between the consumed data resource and another data resource owned by the Data Consumer. In this case, the Data Consumer can perform the interoperability assessment process using the Interoperability Assessment menu of the IDS Connector web application. Figure 27 shows the Interoperability Assessment page with the required inputs to perform the interoperability assessment process.



Request

Step 1: select source schema

Select CSV Source Schema from the list below.
The list shows the IDS Resources from Data Provider that can be accessed by the Data Consumer.

UUID	Title	Description	Keywords	Publisher
http://localhost:8082/a...	1. CSV Source Schema ...	Authors.first_name,Auth...	literature	https://emagizi
http://localhost:8082/a...	2. CSV Source Schema ...	reisnummer,bedrijfsnaa...	logistics	https://emagizi
http://localhost:8082/a...	3. CSV Source Schema ...	trip_id,companies.id,co...	logistics	https://emagizi

Step 2: select target schema

Upload a CSV file that contains the Target Schema.

Target Schema	3_target_schema_nested.csv

Step 3: specify payment information (optional, only for commercial data)

☑ Requires Payment

Payment Model	One-time Payment [✓ Recurring Payment]
Number of Payment Intervals	12
Price per Payment Interval	50

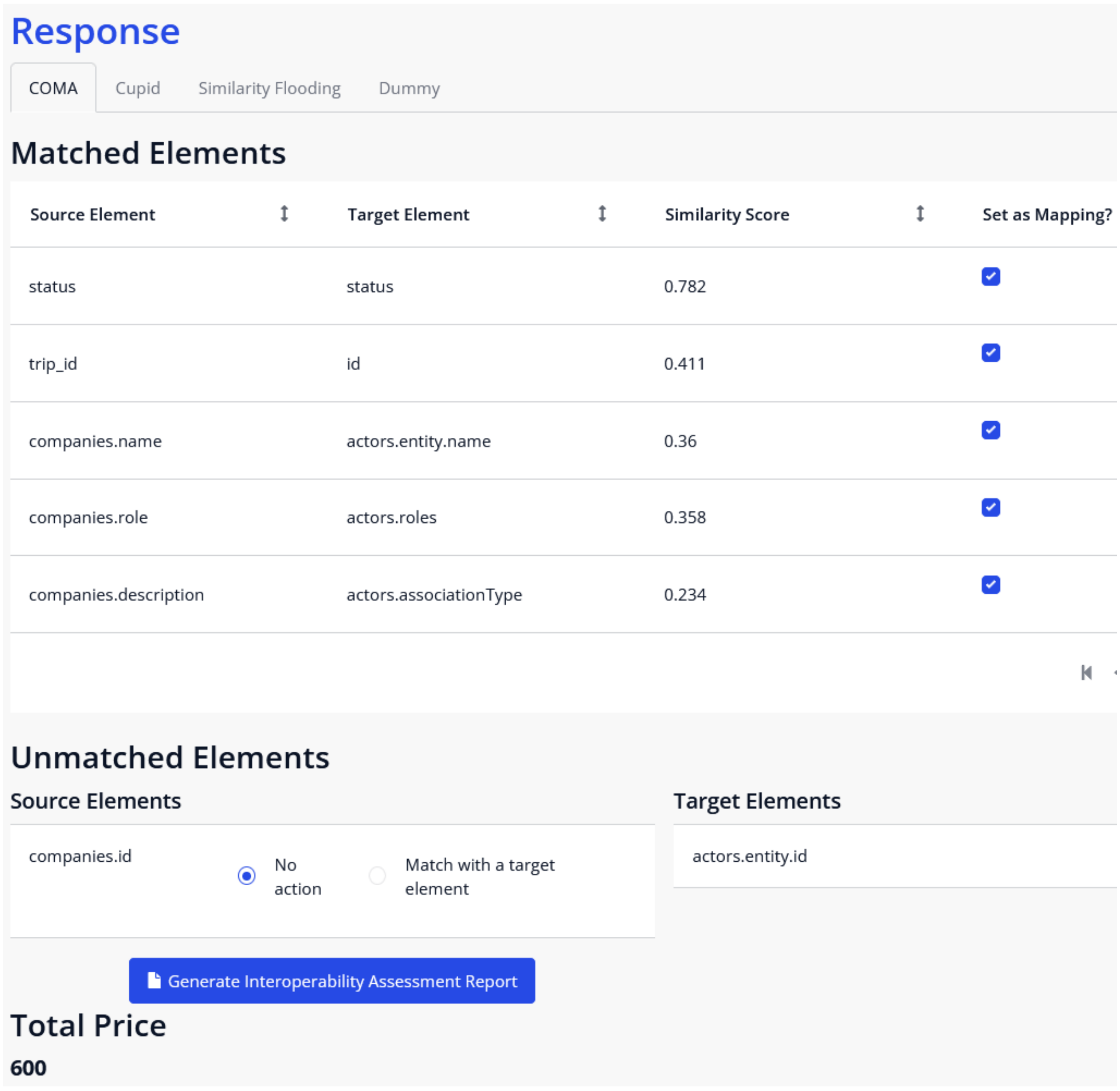
Figure 27 The three inputs of an Interoperability Assessment process

The first step of the interoperability assessment process is to select the source schema from a list of consumed resources. The list of consumed resource is identical to the list in the Data Consumption Overview page depicted in Figure 25. If a specific resource is missing from the list, the Data Consumer needs to initiate a new data consumption process as explained in Section 4.3.3.

After the source schema has been selected, the Data Consumer uploads a target schema that is matched with the source schema. The uploader only accepts CSV as the file type. However, the uploaded CSV file is not validated until the interoperability assessment is performed. The Data Consumer has to make sure that the uploaded file only contains comma-separated table headers that conform to a proper CSV format.

The last input of the interoperability assessment process is the payment information about the offered data resource. The Data Consumer provides the payment information only if commercial data is offered. The payment information consists of the two payment models that can be selected in our prototype, along with the fields that correspond to each payment model. For example, the recurring payment model requires the Data Consumer to provide the number of payment intervals and the price per payment interval. The term *payment interval* is a general term for payment periods, such as weekly, monthly, or annually. This term is used because the actual payment period may vary across data resources, depending on the usage policies set by the Data Provider.

After the Data Consumer provides all the required inputs, the interoperability assessment can be performed by calling the Interoperability Simulator API endpoint described in Table 5. The assessment results are depicted in Figure 28. The results are composed of the assessment result from each Matcher. For each Matcher, the result page is divided into two sections that describe the schema matching result and the pricing calculation result, respectively.



Matched Elements

Source Element	Target Element	Similarity Score	Set as Mapping?
status	status	0.782	✓
trip_id	id	0.411	✓
companies.name	actors.entity.name	0.36	✓
companies.role	actors.roles	0.358	✓
companies.description	actors.associationType	0.234	✓

Unmatched Elements

Source Elements

companies.id [✓ No action] Match with a target element

Target Elements

actors.entity.id

Total Price

600

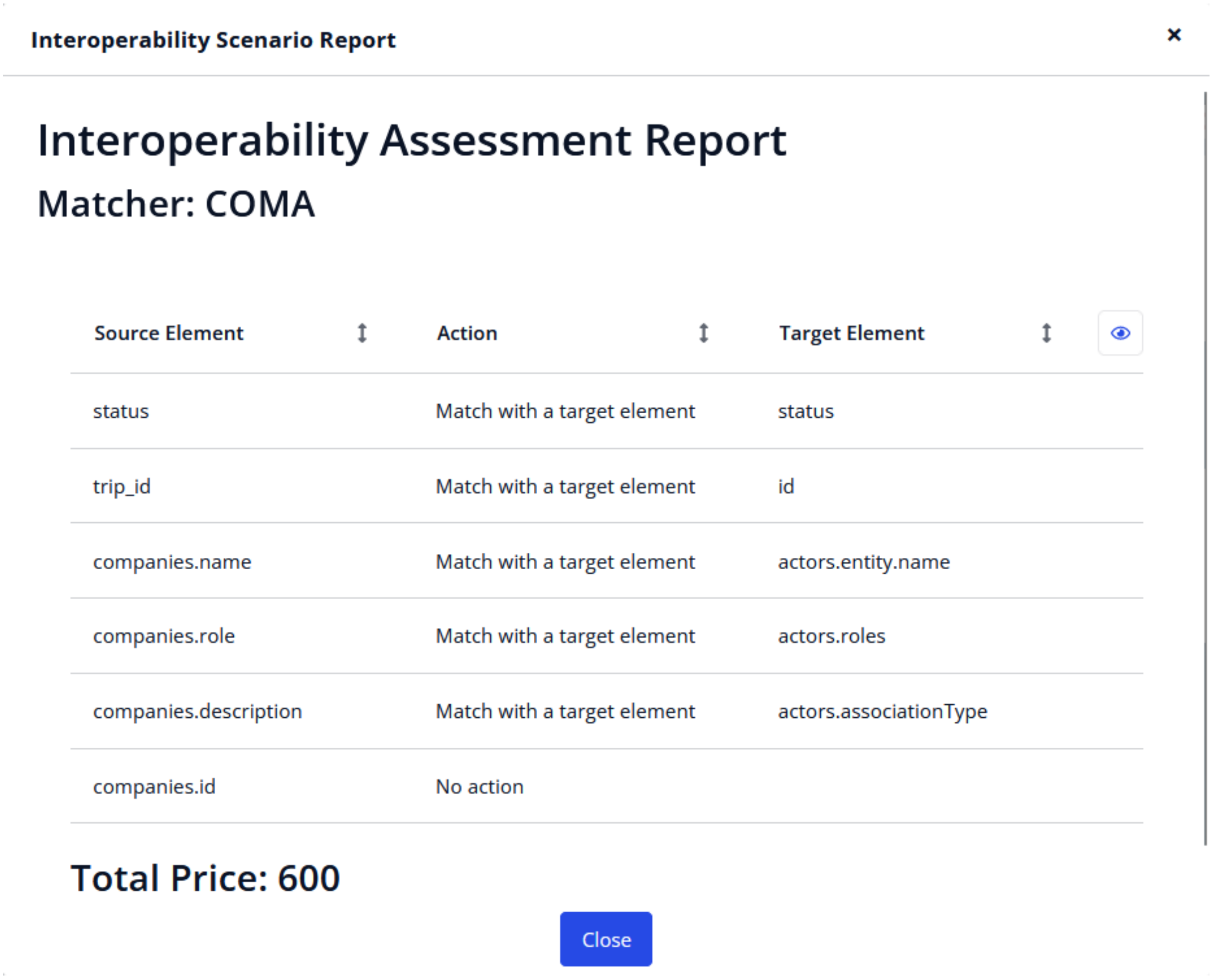
Figure 28 Interoperability assessment results

The schema matching result is further divided into two sections for the matched elements and the unmatched elements. The Matched Elements section contains a table that lists the matched pairs from the source schema to the target schema. Each table row represents a pair of matched elements and its similarity score. The similarity score indicates the syntactic similarity of the matched pair according to the Matcher, having a value range between 0 and 1. A similarity score of 0 indicates no similarity at all, while a similarity score of 1 indicates an identical pair.

The Matched Elements section also contains another column that can be used by the Data Consumer to annotate the schema matching results. For each matched pair, the Data Consumer can decide to accept or reject the matched pair as the correct mapping. By default, the Interoperability Simulator sets the highest similarity score for each source element mapping as the correct mapping. Currently the Interoperability Simulator does not have a threshold for the similarity score as a criterion for a correct mapping. Therefore, it is possible for the Interoperability Simulator to choose a mapping with low similarity score as a correct pair.

The Unmatched Elements section lists all the source schema elements and the target schema elements that do not have matches according to the Matcher. The Data Consumer can set an action to be performed on each unmatched source element. There are two possible actions, either to leave the source element in its current state or to match the source element with another unmatched target element. We use the unmatched source element `companies.id` in Figure 28 as an example: the Data Consumer can choose to either match it with the target element `actors.entity.id`, or to take no action at all.

For each Matcher result, the Data Consumer can generate an interoperability assessment report that shows a summary of the interoperability assessment result. The report summary lists all the source elements that have been set as correct mappings, as well as the remainder source elements that do not have any match with the target elements. An example of an interoperability scenario report is shown in Figure 29.



Interoperability Scenario Report

Interoperability Assessment Report

Matcher: COMA

Source Element	Action	Target Element
status	Match with a target element	status
trip_id	Match with a target element	id
companies.name	Match with a target element	actors.entity.name
companies.role	Match with a target element	actors.roles
companies.description	Match with a target element	actors.associationType
companies.id	No action	

Total Price: 600

Figure 29 Example of interoperability assessment report

4.4 DEPLOYMENT USING CONTAINERIZATION

As discussed in Section 2.3, IDS Connectors and their sub-components are deployed using containerization. The IDS Reference Architecture Model does not prescribe a specific container runtime for the IDS Connectors (IDSA, 2022), however, we chose Docker Engine as the container runtime for this research because it is popular[8] and because it has supporting tools, such as facilities for running multiple containers and a dedicated container registry.

According to the concrete architecture in Figure 17, each application component has a specific set of containers. The IDS Connector application component has two containers that consist of one IDS Connector application container and one database container. In contrast, the Vocabulary Hub application component has five containers to run four Matcher applications and a Vocabulary Hub application that contains the Interoperability Simulator application.

A Docker image is required to run each container. Our prototype uses custom Docker images for most of the containers. Each custom Docker image is built using a collection of commands stored in a Dockerfile. For example, the Vocabulary Hub container uses a custom Docker image named `aldidoanta/vocabulary-hub` that contains a Django application. The code snippet below shows the Dockerfile that defines the `aldidoanta/vocabulary-hub` Docker image.

```
FROM python:3.10.12-slim-bullseye

WORKDIR /vocabulary_hub

ENV PYTHONUNBUFFERED 1
ENV PYTHONPATH /vocabulary_hub:$PYTHONPATH

COPY . /vocabulary_hub/

EXPOSE 8000

RUN pip install -r requirements.txt

CMD ["python", "/vocabulary_hub/manage.py", "runserver", "0.0.0.0:8000"]
```

The custom Docker images are pushed to a public container registry called Docker Hub, under the namespace `aldidoanta`[9]. These Docker images can be downloaded to be reused in future projects, enabling other software engineers to try the custom Docker images in their own development environment.

Most containers of the prototype are built using custom Docker images. The only exception is the database containers that use an existing PostgreSQL Docker image provided by the Docker Hub[10].

The prototype consists of 9 different containers that support different application components and are automatically deployed using Docker Compose, facilitating the deployment process[11]. A Compose YAML file is used as a configuration file in our prototype. The configuration file

[8] https://www.statista.com/statistics/1224618/container-platforms-deployed-runtime/
[9] https://hub.docker.com/u/aldidoanta
[10] https://hub.docker.com/_/postgres
[11] https://docs.docker.com/compose/

contains information such as the container names, the associated Docker images, the port mapping for each container, and the environment variables configuration for each container. Figure 30 shows the container configuration using Docker Compose.

Docker Compose is already sufficient for local deployment of Docker containers, since it only requires a single configuration file and a few command lines to execute the deployment

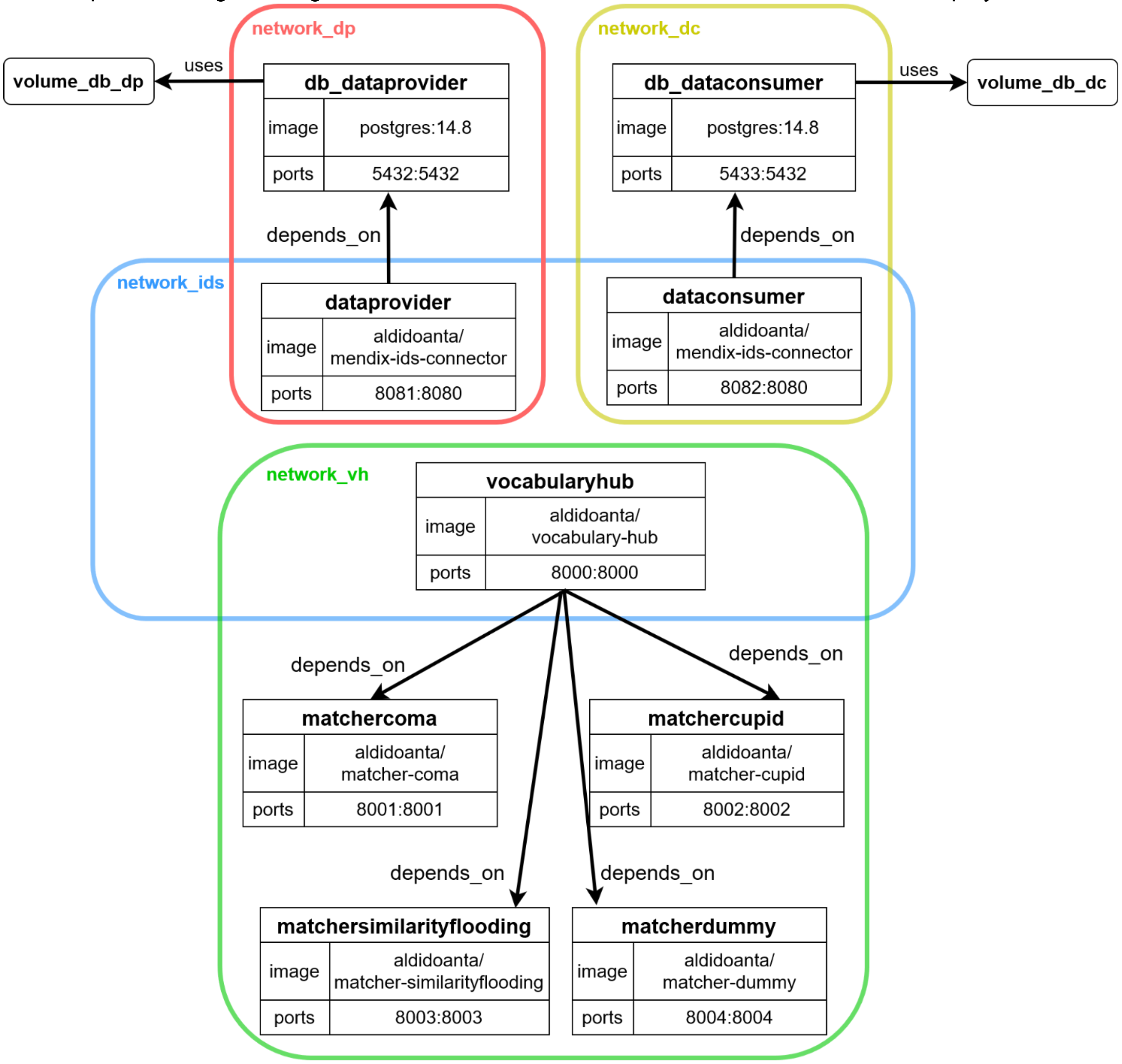


Figure 30 Visual representation of container configuration using Docker Compose

process. However, Docker Compose is not suitable for a production-level deployment that requires fault tolerance and high scalability, since it does not have built-in tools that can support these requirements.

Each time the codebase of a software component is updated, the corresponding Docker image needs to be rebuilt and pushed to the container registry. The entire process requires several command lines that are tedious to execute, especially when rebuilding the Docker image `aldidoanta/mendix-ids-connector` that has another dependency to containerize a

Mendix application[12]. To automate the process of building and publishing the Docker images, the Continuous Integration (CI) approach has been employed.

All artifacts produced in this project, namely the design diagrams, a codebase of the prototype, a slide deck of the research, and this thesis document, are published as two mirror Git repositories hosted on the GitLab server of the University of Twente[13][14].

[12] https://github.com/mendix/docker-mendix-buildpack
[13] https://gitlab.utwente.nl/aldidoanta/mp-isim
[14] https://gitlab.utwente.nl/aldidoanta/mp-isim-frontend

5. INTEROPERABILITY SCENARIOS

This chapter reports on the Treatment Validation task of the design cycle, in which a validation model was developed to validate the solution design. More specifically, this chapter describes how a model of the problem context was developed to support the model of the solution that has been explained in Chapter 4.

5.1 APPROACH

In a validation model, the model of the solution interacts with a model of the problem context. This interaction attempts to simulate a situation where a real-life solution interacts with real-life cases (Wieringa, 2014). In this research, the model of the problem context is the interoperability scenario that can be assessed using the Interoperability Simulator prototype. Each interoperability scenario is composed of the following inputs:

1. A CSV file that represents the source schema offered by the Data Provider
2. A CSV file that represents the target schema uploaded by the Data Consumer
3. An optional pricing information

The main goal of this research has been to design and develop an Interoperability Simulator prototype that can be integrated into an existing data space. In this research, we did not consider measuring the performance of the Matchers. Therefore, this research does not apply a quantitative approach by calculating metrics such as precision and recall against a ground truth.

Three different interoperability scenarios were formulated, each having a unique pair of schemas (source schema and target schema). Each interoperability scenario assessment produces both the pricing calculation result and the schema matching results. The Dummy Matcher is excluded from the schema matching results since it does not perform a proper schema matching operation as its only purpose is to demonstrate that a Matcher using any technology can be integrated with the Interoperability Simulator. The schemas for each scenario have been published on GitHub and can be accessed publicly[15].

The schema-matching result has been divided into three sections, where each section corresponds to the result from a specific Matcher. In a scenario that involves `m` source elements and `n` target elements, the Similarity Flooding Matcher always returns `m x n` matches. The interoperability assesment result is be too long to report if a scenario involves a lot of schema elements. Therefore, the interoperability assessment results presented in this chapter are the condensed version of the actual results.

Each interoperability scenario is discussed in the sequel.

5.2 "ACADEMIC PUBLICATION" SCENARIO

The first scenario uses flat schemas that are usually found in tabular data. The schemas were adapted from the example schemas provided by Valentine's Git repository[16]. The terms in the schema elements are related to academic publications, as displayed in Table 6. The "Academic Publication" scenario simulates a non-commercial data resource. Therefore, pricing information is not considered in this scenario.

[15] https://gist.github.com/aldidoanta/9cdd33787ae90a5401bc0d18957dfb32
[16] https://github.com/delftdata/valentine/tree/master/examples/data

Table 6 Source schema and target schema of the "Academic Publication" scenario

Source Schema
ID
AuthorFirstName
AuthorLastName
CitedBy
Title
Year
DOI

Target Schema
id
author_name
title
cited_by
city
document-type

Pricing Information
None

Table 7 Summary of interoperability assessment result of the "Academic Publication" scenario

COMA				
Matched Elements			**Unmatched Elements**	
Source	**Target**	**Similarity Score**	**Unmatched Source**	**Unmatched Target**
Title	title	0.864	Year	document-type
ID	id	0.858	DOI	city
CitedBy	cited_by	0.719	AuthorFirstName	
AuthorLastName	author_name	0.691		
Cupid				
Matched Elements			**Unmatched Elements**	
Source	**Target**	**Similarity Score**	**Unmatched Source**	**Unmatched Target**
Title	title	1	DOI	document-type
ID	id	1		
CitedBy	cited_by	1		
AuthorLastName	author_name	0.94		
AuthorFirstName	author_name	0.94		
Year	city	0.815		
Similarity Flooding				
Matched Elements			**Unmatched Elements**	
Source	**Target**	**Similarity Score**	**Unmatched Source**	**Unmatched Target**
Title	title	0.276	(none)	(none)
CitedBy	cited_by	0.253		
AuthorLastName	author_name	0.248		
AuthorFirstName	author_name	0.244		
Year	author_name	0.183		
ID	id	0.136		
DOI	id	0.136		

Table 7 presents the summary of interoperabilty assessment results of the "Academic Publication" scenario. According to the table, all Matchers can correctly match schema elements that have identical spelling such as `title`, `id`, and `cited_by`. The Matchers can match these words even though they are written in different case and different order in each schema.

However, the Matchers show different behaviors when attempting to find a match for `author_name`. Cupid and Similarity Flooding match both `AuthorFirstName` and `AuthorLastName` with `author_name`, while COMA only match `AuthorLastName` with `author_name`. Moreover, Cupid and Similarity Flooding make a mistake by matching `Year` with another target element, while `Year` actually does not have any matches in the target schema. The matches made by Similarity Flooding also have low similarity scores compared to the matches made by the other Matchers.

5.3 "DUTCH TO ENGLISH" SCENARIO

The second scenario attempts to simulate a situation in the Dutch logistics domain, in which the Data Provider provides a schema in Dutch, while the Data Consumer needs to use the schema in English. This "Dutch to English" scenario uses flat schemas that are common in tabular data, similar to the previous scenario. The difference from the previous scenario is that this scenario uses schema elements that are related to the logistics domain, as presented in Table 8. This scenario also simulates a commercial data resource that requires the Data Consumer to perform a one-time payment before accessing the actual data.

Table 8 Source schema, target schema, and pricing information of the "Dutch to English" scenario

Source Schema
reisnummer
bedrijfsnaam
omschrijving
land
adres
telefoonnummer

Target Schema
trip_number
company_name
company_profile
company_adress
company_status
company_phone
company_email

Pricing Information	
Payment Model	Price
One-time Payment	100

Table 9 Interoperability assessment result of the "Dutch to English" scenario

COMA				
Matched Elements			Unmatched Elements	
Source	Target	Similarity Score	Unmatched Source	Unmatched Target
adres	company_adress	0.355	reisnummer	company_name
telefoonnummer	trip_number	0.308	omschrijving	company_phone
land	company_email	0.207	bedrijfsnaam	company_status
				company_profile

Cupid				
Matched Elements			Unmatched Elements	
Source	Target	Similarity Score	Unmatched Source	Unmatched Target
adres	company_adress	0.729	reisnummer	trip_number
land	company_status	0.791	omschrijving	
land	company_name	0.787	telefoonnummer	

land	company_email	0.784	bedrijfsnaam	
land	company_profile	0.752		
land	company_phone	0.75		
land	company_adress	0.733		
Similarity Flooding				
Matched Elements			**Unmatched Elements**	
Source	**Target**	**Similarity Score**	**Unmatched Source**	**Unmatched Target**
reisnummer	trip_number	0.249	(none)	(none)
adres	company_adress	0.225		
telefoonnummer	trip_number	0.217		
telefoonnummer	company_name	0.201		
bedrijfsnaam	trip_number	0.194		
land	company_adress	0.192		
omschrijving	company_phone	0.189		
adres	company_profile	0.186		
omschrijving	company_status	0.174		
omschrijving	company_email	0.176		

Total Price	100

Table 9 shows the results of the "Academic Publication" scenario. The price calculation result gives the correct amount of the total price. However, all Matchers have difficulties in matching the schema elements. COMA and Similarity Flooding give low similarity scores for their matches, while Cupid gives a higher similarity score even though most of the matches are incorrect.

The reason behind this behavior is that the Matchers do not have a dedicated dictionary to translate Dutch words to English words, and vice versa. COMA and Cupid rely on linguistic approaches that are limited to English word. Therefore, the Matchers can only detect `adres - company_adress` as the correct match because the two elements have similar structure on letters in both languages.

5.4 "NESTED ELEMENTS" SCENARIO

The third scenario simulates a situation where a Data Consumer needs to transform an arbitrary schema into another schema that uses a standard data model. In this scenario, the source schema is loosely based on a logistic-domain data model named The Open Trip Model (OTM)[17]. Therefore, some of the target schema elements use logistics-related terms such as `trip`. The data model specified in OTM is formatted in JSON, while the current prototype only supports CSV table headers. Therefore, a workaround was applied by flattening the nested JSON structure using a dot ('.') delimiter.

This scenario also simulates a commercial data resource that requires the Data Consumer to perform recurring payments. Table 10 describes the inputs of the "Nested Elements" scenario, which consist of the source schema, the target schema, and the pricing information.

[17] https://www.opentripmodel.org/

Table 10 Source schema, target schema, and pricing information of the "Nested Elements" scenario

Source Schema
trip_id
companies.id
companies.name
companies.description
companies.role
status

Target Schema
id
status
actors.entity.id
actors.entity.name
actors.roles
actors.associationType

Pricing Information		
Payment Model	**Payment Intervals**	**Price per Payment Interval**
Recurring Payment	12	50

Table 11 Interoperability assessment result of the "Nested Elements" scenario

COMA				
Matched Elements			**Unmatched Elements**	
Source	**Target**	**Similarity Score**	**Unmatched Source**	**Unmatched Target**
status	status	0.782	companies.id	actors.entity.id
trip_id	id	0.411		
companies.name	actors.entity.name	0.36		
companies.role	actors.roles	0.358		
companies.description	actors.associationType	0.234		
Cupid				
Matched Elements			**Unmatched Elements**	
Source	**Target**	**Similarity Score**	**Unmatched Source**	**Unmatched Target**
status	status	1	companies.name	actors.entity.name
trip_id	id	0.84	companies.id	actors.associationType
			companies.description	actors.entity.id
			companies.role	actors.roles
Similarity Flooding				
Matched Elements			**Unmatched Elements**	
Source	**Target**	**Similarity Score**	**Unmatched Source**	**Unmatched Target**
status	status	0.319	(none)	(none)
companies.role	actors.roles	0.251		
companies.name	actors.entity.name	0.241		
companies.description	actors.associationType	0.232		
companies.id	actors.entity.id	0.23		
trip_id	id	0.22		

Total Price	600

Table 11 shows the results of the “Academic Publication” scenario. The calculated total price is correct can be verified by multiplying the number of payment intervals by the price per payment interval. Compared to the previous scenario that uses a language other than English, the Matchers in this scenario display a better performance by correctly matching most of the schema elements. The only exception is the Cupid Matcher that only matches two pairs of schema elements, while discarding the other pairs as unmatched elements. However, the Similarity Flooding Matcher still shows the same behavior by giving low similarity scores to its matches.

6. PROTOTYPE VALIDATION

After the potential solution has been developed as a prototype, the next step of the Treatment Validation task of the design cycle is to validate the proposed solution. This is the last part of the Treatment Validation task, in which the solution prototype is validated against the intended goals to solve the identified interoperability problems. Section 6.1 describes the approach taken to validate the solution, followed by Section 6.2 that reports the results of the validation approach.

6.1 APPROACH

This research uses the expert opinion approach to validate the prototype (Wieringa, 2014). Several experts from both the academia and the industry were interviewed for their opinions about the functionalities of the Interoperability Simulator. They were contacted via email to schedule the validation interview sessions, which were conducted either in-person and online. Table 12 lists the participating experts and their affiliations. An ID is used to refer to each specific expert when explaining the validation results in Section 6.2. Three researchers from TNO are considered as one single expert (E4), since all of them were interviewed in one meeting and the meeting notes did not separate the opinions given by each individual.

Table 12 Experts participating in the validation interview sessions

ID	Role	Affiliation
E1	Researcher and PhD candidate	Department of Industrial Engineering and Business Information Systems (IEBIS), University of Twente (UT)
E2	Expert Services	eMagiz B.V.
E3	Expert Services	eMagiz B.V.
E4	1. Senior Business Consultant - Data Ecosystem 2. Scientist Innovator 3. Senior Advisor - Data Sharing and Interoperability	TNO

A validation interview session began with a short presentation about the Interoperability Simulator and the overview of its architecture. The presentation was followed by a prototype demonstration using one of the interoperability scenarios discussed in Chapter 5. The expert played the role of a Data Consumer who wanted to assess the information gap wih a data asset from the Data Provider. After that, a discussion was conducted to elicit the opinion from the experts using several questions as discussion pointers.

The validation questions have been derived from the domain-level requirements of the Software Requirements Specification, which is described in Section 3.2. Table 13 shows the formulated validation questions based on the domain-level requirements. Domain-level requirement DoR4 is not included even though it is related to Data Provider's tasks, because the requirement has not been fulfilled by the current prototype.

Table 13 The questions for the prototype validation interview that are derived from the domain-level requirements

Req-Ref	Domain-Level Requirement	Q-Ref	Validation Question
DoR1	The Interoperability Simulator should help the Data Consumer gather the required information to assess the information gap.	Q1	Is the gathered information by the Interoperability Simulator already sufficient to assess the information gap in a data space? To what extent?
DoR2	The Interoperability Simulator should support the Data Consumer in assessing the information gap using the gathered information.	Q2	Are the schema matching results and the cost calculation result of the Interoperability Simulator helpful enough for the Data Consumer to assess the information gap of the interoperability scenario? To what extent?
DoR3	The Interoperability Simulator should be able to report the assessed information gap to the Data Consumer.	Q3	Is the interoperability assessment report helpful enough for the Data Consumer to understand the required actions to follow up on the interoperability scenario? To what extent?

6.2 VALIDATION RESULTS

This section reports on the expert opinions that were elicited during the validation interview sessions. The opinions are grouped by validation question and are outlined in the following sections. An additional section is also added to report expert opinions that are not directly related to the three validation questions. Moreover, the notes taken during the validation interview sessions are compiled in Appendix A.

6.2.1 Validation Question Q1

Is the gathered information by the Interoperability Simulator already sufficient to assess the information gap in a data space? To what extent?

The experts stated that the inputs for the Interoperability Simulator prototype are already sufficient. The displayed user interface for the inputs is logical and straightforward. However, they agreed that the current inputs are possible under the assumption that the Data Consumer has already found the Data Provider. If the Data Consumer still needs to discover and find the Data Providers who are willing to publish their schemas, the inputs of the prototype need to be revised.

If the prototype uses a standard data model in the interoperability scenario, E1 suggested to explicitly indicate the usage of the data model, either in the source schema or in the target schema. Regarding the user interface, E2 recommended an updated user interface that represents the source schema and the target schema as tables, with the schema elements as the table rows. E3 mentioned that the current prototype requires the Data Consumer to upload a new target schema for each scenario. It would be more efficient if the Data Consumer is allowed to reuse the target schemas that have been uploaded before.

E2 and E3 had differing opinions on the pricing information. E2 argued that the payment information should be provided by the Data Provider, instead of being provided manually by the Data Consumer. The payment information might be added as additional metadata in the data resource's usage policy. On the other hand, E3 argued that the payment information should be asked after the interoperability assessment report, because the interoperability of the schema elements has a higher priority than calculating the total price. However, E3 also agreed with E2 in that the pricing information can also be part of the source schema's metadata.

E4 did not make significant remarks regarding the Interoperability Simulator inputs. They suggested to improve the prototype by handling syntactic metadata such as the data types of the schema elements and the value constraints for each schema element. They argued that these syntactic metadata would be useful to improve the prototype performance on the Schema Matching function.

6.2.2 Validation Question Q2

Are the schema matching results and the cost calculation result of the Interoperability Simulator helpful enough for the Data Consumer to assess the information gap of the interoperability scenario? To what extent?

Each expert answered this question by pointing out a specific aspect. E1 made remarks about how the prototype can compare schemas and show the similarity scores for each mapping. E1 added that it would be better if the prototype displayed a general interoperability score to the Data Consumer. In this way, the Data Consumer would have a general idea about how interoperable the scenario is, and the Data Consumer would be able to make adjustments to the data schema. E1 also commented about the scalability issue that needs to be addressed by the prototype, once it needs to handle big schemas. However, scalability was out of the scope of this research. Instead, it has been considered as one of the topics for future work.

E2 argued that a data type definition for each schema element is important to improve the schema matching results. However, E2 also understood the technical limitations of the current prototype. This data type definition would be useful for the next iteration of the prototype, in which the data instances are also used to perform the schema matching operation. E2 also argued that the Data Consumer should be able to add some notes about the required data transformation for each matched pair.

Similar to E1, E3 also considered the similarity score on the interoperability assessment results as useful information to display. However, E3 also gave feedback about the visual representation of the assessment results. The differences between Matchers should be highlighted by the prototype to help the Data Consumer decide on the best Matcher to use.

E4 suggested that the prototype should have used context-aware capabilities for its Schema Matching function. Examples of context would be the data types of the schema elements, or domain-specific context from IDS Vocabulary Hub that can help match the elements from the two schemas. Moreover, the interoperability assessment results might be confusing for the user since they only consist of schema matching results and the similarity scores. The prototype should interpret the meaning of the results and explain it to the user, so the user can understand the interoperability assessment results.

6.2.3 Validation Question Q3

Is the interoperability assessment report helpful enough for the Data Consumer to understand the required actions to follow up on the interoperability scenario? To what extent?

E1 and E3 had a similar opinion about the interoperability assessment report. E1 said that the prototype would be more useful if the Data Consumer could specify a list of mandatory and non-mandatory schema elements. This mandatory and optional specification is necessary because the Data Consumer needs to know if the mandatory requirements for the scenario have been fulfilled. E3 commented that the Data Consumer needs to know if the schema matching results have covered all the required elements, either from the source elements or from the target elements. This schema matching coverage might help the Data Consumer to understand how interoperable the source schema and the target schema are. E3 also suggested the usage of dots and arrowed lines to improve the assessment report visualization.

E2 followed up on the answer to Validation Question Q2. In the assessment report, E2 suggested to add another column that describes the transformation steps for each matched element pair. If this functionality is added, this prototype has more potential to be used as a semi-automatic schema matching tool that can complement eMagiz current tools.

E4 approved the information representation on the interoperability assessment report. However, they also argued that data interoperability is a complex problem that cannot be naively solved using a single software solution. The Schema Matching function of the prototype is not the final solution to solve all the challenges of data interoperability. Therefore, they recommended to treat the prototype as a helper tool rather than as the final solution. As a helper tool, the Interoperability Simulator could give the initial results of the interoperability scenario that can be understood by the users.

6.2.4 Other Remarks

E2 recommended two approaches to present the Interoperability Simulator idea: to explain the Interoperability Simulator functionalities as they are, similar to what has been done in the validation interviews, and to present the Interoperability Simulator functionalities and mention the potential real-life cases that can be solved by the simulator. For example, a hypothetical scenario can be presented in which IDS becomes a standard in the European Union. In this scenario, the Interoperability Simulator can play a role to make data integration processes more efficient.

E4 are familiar with the IDS business processes. Therefore, they gave feedback about the position of the Interoperability Simulator inside the IDS business processes. They commented that the Simulator should belong in the Data Exchange process rather than in the Use IDS App process. They also argued that the Pricing Calculation function does not fit well with the current architecture of the prototype. They suggested to use the Pricing Calculation function as a separate function that works at the business/legal interoperability level.

E4 concluded the validation interview by mentioning potential ideas that can improve the Schema Matching function of the prototype. One idea is to use domain-specific data together with large language models such as Generative Pre-trained Transformers (GPT). Another idea is to use federated learning solutions with CSV files, which is another semi-supervised schema matching approach similar to what has been demonstrated with the prototype.

7. FINAL REMARKS

This chapter concludes the research on the Interoperability Simulator that is reported in this thesis. Section 7.1 outlines the conclusions of the research, and Section 7.2 points out the limitations of the current design and prototype of the Interoperability Simulator, along with the topics for future work.

7.1 CONCLUSIONS

This section gives the conclusions of this research by answering the research sub-questions that have been formulated in Section 1.2. Each research sub-question is answered in the sequel.

7.1.1 Research Sub-questions RSQ1

What attributes can be gathered by an Interoperability Simulator to assess the information gap in a data interoperability scenario? What are the limitations?

The Interoperability Simulator aims to solve an information gap in a data interoperability scenario, whose definition has been defined in Section 3.1. The definition serves as one of the references to formulate the software requirements specification, which is subsequently used to design a reference architecture for the Interoperability Simulator. According to the reference architecture specified in Section 3.3, the inputs of an Interoperability Simulator consist of the provided schema from the Data Provider, the requested schema provided by the Data Consumer, and the optional pricing information that is also provided by the Data Consumer.

The Schema Matching function of the Interoperability Simulator accepts the source schema from the Data Provider and the target schema from the Data Consumer. Each schema has a representation of a CSV string that contains table column names as schema elements. Besides the name of the schema elements, no additional information is provided by the schema. Moreover, the data instances of the schema elements (the table column names) are also not taken into account in the Schema Matching function.

The Pricing Calculation function accepts the pricing information that is filled in by the Data Consumer. We implemented two different payment models, namely the one-time payment model and the recurring payment model. Each payment model has slightly different pricing information that must be provided by the Data Consumer. However, the pricing information in our prototype is very limited and does not reflect the pricing information of commercial data in real-world cases. Additionally, an expert made a remark that the payment information should be provided directly by the Data Provider (see Section 6.2.1).

7.1.2 Research Sub-questions RSQ2

Using the gathered attributes from the Data Provider and the Data Consumer, how does an Interoperability Simulator assess the information gap?

As described in Section 4.2, our Interoperability Simulator consists of the Schema Matching function and the Pricing Calculation function. Each function is responsible for a specific calculation and accepts some specific inputs. The Schema Matching function uses the source schema and the target schema to perform schema matching operations using Matchers. In the reference architecture presented in Section 3.3.2, each Matcher is deployed as a separate container from the Interoperability Simulator's container. The design allows the data space maintainer to add new Matchers or remove available Matchers, decoupling the Matchers implementation from the internal workings of the Interoperability Simulator.

The Pricing Calculation function uses the pricing information provided by the Data Consumer to calculate the total price for the data resource. As mentioned in Section 4.2.2, the current prototype employs a simple multiplication operation to calculate the total cost, depending on the selected payment model. Moreover, expert suggested to move the Pricing Calculation function to outside the Interoperability Simulator prototype, as this function is more suitable to solve legal and business interoperability issues than syntactic interoperability issues.

7.1.3 Research Sub-questions RSQ3

How does an Interoperability Simulator report the assessed information gap to the interacting organizations?

In our Interoperability Simulator prototype, the results from the Schema Matching function and the Pricing Calculation function are combined as the interoperability assessment results. The results are composed of the schema matching results from each Matcher that are grouped by the matched elements and the unmatched elements. The assessment results allow the Data Consumer to annotate the schema matching results, both on the matched and the unmatched elements. The report also shows the total costs of the data resource, in case pricing information is provided by the Data Consumer.

The annotated schema matching results and the information about the total cost can be summarized into a single Interoperability assessment report, as depicted in Figure 29. The report presents the mapping results from the source schema to the target schema, along with the action that needs to be performed for each source element.

7.2 LIMITATIONS AND FUTURE WORK

This section discusses the limitations of our research from different perspectives and the potential topics for future work.

7.2.1 Design of the Interoperability Simulator

Regarding the use case of the Interoperability Simulator, there are two intended use cases: to assess interoperability when a Data Consumer requests data to a Data Provider, and to assess interoperability when a Data Provider wants to standardize its data resource using a standard data model. This prototype focuses on the first use case, since it is more general and can be applied to any data space, even to a data sharing environment that is not based on the data space concepts. The second case is potential to be realized in a data space where standards are important. For example, the “Nested Elements” scenario used in Section 5.4 can form a use case that involves the Vocabulary Hub as the provider of the standard data model.

Figure 12 shows that our Interoperability Simulator has two main business functions. The first business function (interoperability assessment) has been realized by the design and prototype of this research. However, the current prototype still lacks the functionality to follow up on the interoperability assessment results and to recommend the actual interoperability solutions. This missing functionality is represented as a second business function in Figure 12, which is out of scope of this research. Additional research can be carried out to investigate this second business process, to realize a more complete Interoperability Simulator for data spaces.

7.2.2 Input Data

Section 3.1 mentions that an interoperability scenario in an IDS environment happens during the Data Offering step. Therefore, an interoperability scenario in IDS can only be solved using schema-level matchers because the Data Consumer has not gained access to the actual data instances. This research uses CSV table headers to represent schemas because the Matchers only accepts string-based schema elements without additional metadata such as data type and type value constraints. These limitations are reflected in the results of different interoperability scenarios presented in Chapter 6.

To improve the schema matching performance of the Matchers, there are some options: (1) to involve data instances in the schema matching operation, the three instance-level Matchers in the Valentine implementation can be used as starting point. (2) to use another schema format that supports metadata, such as JSON schema or XML schemas that use XML Schema Definition (XSD). These two options can help the Matchers differentiate similar schema elements based on the differences in their data instances and metadata. However, the two options require changes to the business process of the Interoperability Simulator, as well as on the implementation logic of the Matchers.

7.2.3 Matchers

As mentioned in Section 7.1.2, the Matchers are designed to be loosely coupled from the Interoperability Simulator. However, the Matchers integration with the Interoperability Simulator is still hardcoded by manually listing the list of the Matchers inside the Interoperability Simulator codebase. Adding or removing a new Matcher requires minor changes to the Interoperability Simulator source code. Alternatively, it is possible to use service discovery that can automatically detect changes to the available Matchers. The service discovery mechanism does not require the entire system to be restarted, making it a suitable addition to production-ready solutions.

This research aims to demonstrate the technical feasibility of an Interoperability Simulator for data spaces. Therefore, this research started at the most technical interoperability layer, which is the syntactic interoperability layer. In a next iteration of this research, one can consider the semantic level, utilizing context-aware measures such as using domain-specific vocabularies. By using vocabularies, the Vocabulary Hub of the current prototype can have an actual functionality rather than just act as a container for the Interoperability Simulator. Research should be carried out to investigate the possibility of using domain-specific vocabularies for schema-matching operations in a domain-specific data space.

REFERENCES

Database Group Leipzig. (n.d.). *COMA 3.0 | Database Group Leipzig*. Retrieved September 21, 2023, from https://dbs.uni-leipzig.de/Research/coma.html

Do, H.-H., & Rahm, E. (2002). COMA — A system for flexible combination of schema matching approaches. In *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases*. https://doi.org/10.1016/b978-155860869-6/50060-3

Firdausy, D. (2023). *Designing Essential Components For Logistics Data Spaces: Connecting Logistics interfaces, Converters, Knowledge, and Standards* [University of Twente]. https://doi.org/10.3990/1.9789036557177

Franklin, M., Halevy, A., & Maier, D. (2005). From databases to dataspaces: A new abstraction for information management. *SIGMOD Record*, *34*(4). https://doi.org/10.1145/1107499.1107502

Gewirtz, D. (2016, February 11). *Billion-dollar mistake: How inferior IT killed Target Canada*. ZDNET. https://www.zdnet.com/article/billion-dollar-failures-how-bad-decisions-and-poor-it-killed-target-canada/

Halevy, A., Franklin, M., & Maier, D. (2006). Principles of dataspace systems. *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. https://doi.org/10.1145/1142351.1142352

IDSA. (n.d.). *International Data Spaces*. Retrieved April 4, 2023, from https://internationaldataspaces.org/

IDSA. (2022). *International-Data-Spaces-Association/IDS-RAM_4_0*. https://github.com/International-Data-Spaces-Association/IDS-RAM_4_0

IEEE. (1991). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. IEEE Press. https://doi.org/10.1109/IEEESTD.1991.106963

Ionescu, A. (2020). *Reproducing state-of-the-art schema matching algorithms* [Delft University of Technology]. http://resolver.tudelft.nl/uuid:9f8056e6-cfdf-4240-99e3-5f45947d1fa7

Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A., & Katsifodimos, A. (2021). Valentine: Evaluating matching techniques for dataset discovery. *Proceedings - International Conference on Data Engineering*, *2021-April*. https://doi.org/10.1109/ICDE51399.2021.00047

Lauesen, S. (2002). Software Requirements: Styles & Techniques: Styles and Techniques. In *Pearson Education, Essex*.

Madhavan, J., Bernstein, P. A., & Rahm, E. (2001). Generic schema matching with cupid. *VLDB 2001 - Proceedings of 27th International Conference on Very Large Data Bases*.

Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. *18th International Conference on Data Engineering (ICDE'02)*. https://doi.org/10.1109/ICDE.2002.994702

Nagarajan, M., Verma, K., Sheth, A. P., & Miller, J. A. (2007). Ontology driven data mediation in Web services. *International Journal of Web Services Research*, *4*(4). https://doi.org/10.4018/jwsr.2007100105

Network of the National Library of Medicine. (n.d.). *Data Interoperability*. Retrieved March 21, 2023, from https://www.nnlm.gov/guides/data-glossary/data-interoperability

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, *10*(4). https://doi.org/10.1007/s007780100057

Sheth, A. (1998). Changing Focus on Interoperability in Information Systems:From System, Syntax, Structure to Semantics. In *Interoperating Geographic Information System*. https://doi.org/10.1007/978-1-4615-5189-8_2

Sheth, A., & Kashyap, V. (1992). So far (schematically) yet so near (semantically). In *IFIP Transactions A: Computer Science and Technology* (Issue A-25). https://doi.org/10.1016/b978-0-444-89879-1.50022-1

The Open Group. (2022). ArchiMate 3.2 Specification. In *Opengroup.org*. https://pubs.opengroup.org/architecture/archimate3-doc/ch-Definitions.html

Wegner, P. (1996). *Interoperability*.

Wieringa, R. J. (2014). Design science methodology: For information systems and software engineering. In *Design Science Methodology: For Information Systems and Software Engineering*. https://doi.org/10.1007/978-3-662-43839-8

APPENDIX A. INTERVIEW NOTES

Expert: Researcher and PhD candidate at IEBIS-UT
Date: 12 September 2023

Is the gathered information by the Interoperability Simulator already sufficient to assess information gap in a data space? To what extent?

Yes, it is sufficient. The approach is straightforward. I assume the Data Consumer already has knowledge about IDS and has been onboarded to the data space. And I assume the Data Consumer has already found the Data Provider, so the required metadata have been present in the data space. However, if you want to use standard data models such as OTM, you need to indicate that the Interoperability Simulator is using those data models as the source or the target data.

Are the schema matching results and the cost calculation result of the Interoperability Simulator helpful enough for the Data Consumer to assess the information gap of the interoperability scenario? To what extent?

I like that the prototype can compare schemas, and that it shows the similarity scores for the schema mappings. It would be nice to visualize the mapping like how eMagiz does in their software product, so the user can see the schema mapping results and the relationship between the schema elements better.

For small schemas, this prototype would do enough. However, it needs to have a better representation to handle big schemas. For example, by having a bigger screen, or only showing mapping possibilities for one specific schema element. For big schemas, a Data Consumer might only be interested in a few schema elements instead of the entire schema elements.

Moreover, it would be nice to have a general interoperability score, to show how interoperable the scenario is. It will help the Data Consumer assess the interoperability scenario and make adjustments to the target schema.

Is the interoperability assessment report helpful enough for the Data Consumer to understand the required actions to follow up on the interoperability scenario? To what extent?

As I said before, it would be good to put the general interoperability score in the report. It would be better to have a list of mandatory and non-mandatory elements. These mandatory and non-mandatory elements might be specified by the Data Consumer at the beginning of the interoperability scenario. This is necessary because the Data Consumer needs to know if the mandatory requirements have been fulfilled.

Expert: First Expert Services at eMagiz B.V.
Date: 14 September 2023

Is the gathered information by the Interoperability Simulator already sufficient to assess information gap in a data space? To what extent?

Regarding user interface, the source and the target schemas should be represented as tables. The source schema as one table, and the target schema as another table. This way, it would help the user to check the inputs. Moreover, the payment information should be provided by the Data Provider instead of being provided manually by the Data Consumer.

Are the schema matching results and the cost calculation result of the Interoperability Simulator helpful enough for the Data Consumer to assess the information gap of the interoperability scenario? To what extent?

It is essential to have a data type definition for the schema, but I can understand the technical limitations because you are only using CSV table headers. This data type definition is essential to differentiate the CSV fields, especially if you are using matcher algorithms that take into account the data instances. There are real-life cases where the two fields have exactly the same name (such as `ID`), but they use different data type (for example one uses string, while the other one uses integer).

It would also be useful to add options about data transformation, about what exactly should be done for each matched pair. For example, there is a match for the field `Country`. One schema might use two-letter country code, while the other schema uses three-letter country code. It would be useful if the user can add some notes about the required data transformation for this case.

Is the interoperability assessment report helpful enough for the Data Consumer to understand the required actions to follow up on the interoperability scenario? To what extent?

You can add another column that shows the description of the transformation step that we discussed in the previous question. This interoperability report is something that can be used before the actual integration step happens, like what we do at eMagiz. This prototype is potential to be used for semi-automated matching that can help eMagiz users with their schema matching tasks, which are usually done manually by the users.

Other remarks

There are two directions for your thesis pitch. The first one is by explaining about the prototype functionalities like what you just did here, which is a straightforward approach. The second one is by also mentioning the added value or the potentials for the real-life cases, which would be more interesting. For example, imagine if IDS becomes a standard in the European Union, and every organization needs to interact with each other through IDS. The Interoperability Simulator can play a role in this situation to make data integration processes more efficient.

Expert: Second Expert Services at eMagiz B.V.
Date: 18 September 2023

Is the gathered information by the Interoperability Simulator already sufficient to assess information gap in a data space? To what extent?

The prototype design and the displayed information is quite logical. Regarding the target schema, it might be useful if the Data Consumer can also choose from a list of the previously uploaded target schemas. That would make the process more efficient since the Data Consumer might use the same schema for different interoperability scenarios.

The payment information input should be asked after the final interoperability report, because the Data Consumer needs to focus on the schema-matching results first, rather than worrying about the price to access the actual data. Otherwise, as the other expert suggested in the previous interview, the pricing information should be part of the metadata of the source schema. It is the responsibility of the Data Provider to provide the pricing details for their own data.

Are the schema matching results and the cost calculation result of the Interoperability Simulator helpful enough for the Data Consumer to assess the information gap of the interoperability scenario? To what extent?

Yes. It is good that you put the similarity scores on the report. Maybe you can improve the visualization of the schema matching results or show how the schema elements were matched in a more visual way. Currently your schema matching results are grouped by the Matchers. However, the user needs to understand the differences between the Matchers, so they can choose the best Matcher for their use case. You should highlight the differences between the schema-matching results to help the user spot the differences and decide the best Matcher for their scenario.

Is the interoperability assessment report helpful enough for the Data Consumer to understand the required actions to follow up on the interoperability scenario? To what extent?

As a Data Consumer, you need to make sure that the requirements you set for your target elements have been fulfilled. You need to know how many of the source elements that have been matched with the target elements. These numbers might help you as the end-user to quickly understand how interoperable the source schema and the target schema are. You can also give a general interoperability score on the report.

To improve the user interface of the report, you can create a representation similar to eMagiz's product, where you can use dots and arrowed lines to show the relationship between the source schema and the target schema. If possible, you can also have a single report combined from different Matchers. It would help the Data Consumer read all the interoperability assessment results instead of having to generate a report for each Matcher.

Expert: Three researchers at TNO
Date: 26 September 2023

Is the gathered information by the Interoperability Simulator already sufficient to assess information gap in a data space? To what extent?

Currently it is already sufficient. However, it would be better if the Simulator can handle more syntactic metadata such as JSON attributes along with the data types and the constraints.

Are the schema matching results and the cost calculation result of the Interoperability Simulator helpful enough for the Data Consumer to assess the information gap of the interoperability scenario? To what extent?

It would be good if context-aware results could be provided by the Matchers. The schema context would improve the performance of the interoperability assessment results. The context could be anything, such as data types (that was mentioned in the previous question) or domain-specific context that can give more meaning to the schema elements. This functionality to involve schema context would be a promising future work for this Interoperability Simulator.

It would also be good if the Simulator can help the end-user understand the results by interpreting the meaning of the interoperability assessment results. Currently it only shows the results between the Matchers, without giving a user-friendly explanation about the results.

Is the interoperability assessment report helpful enough for the Data Consumer to understand the required actions to follow up on the interoperability scenario? To what extent?

We like your approach on how to present the information in the Interoperability Assessment Report. However, data interoperability is a complex problem. We cannot be too naive in how we approach the problem. A schema matching solution is not the final solution to solve data interoperability problem. Therefore, in an interoperability scenario, we would position your Interoperability Simulator as a “quick scanner” to show an overall result rather than a detailed result. We recommend you to focus on this high-level, overall result.

Other remarks

Regarding Interoperability Simulator’s position in the business processes of IDS, the Simulator should belong in the Data Exchange process rather than in the other processes. Moreover, the Pricing Calculation function does not fit in the Interoperability Simulator. We suggest removing the Pricing Calculation function from the Interoperability Simulator and use it as a separate function that works at the business/legal level of interoperability. The Schema Matching function of the Simulator can still fit at the semantic/technical level of interoperability.

There are other promising approaches that can be used to improve the schema-matching results. Domain-specific training data can be trained on large language models such as GPT, which can be used as a Matcher. There is also this idea of using federated learning solutions with CSV files. It might not solve the interoperability problem completely, but you can correct the errors made by the automatic Matcher.