QUARRIABLE SMART CITY INTERNET OF THINGS DATA MARKETPLACES

Naeima Hamed
School of Computer Science and Informatics
Cardiff University, UK
hamednh@cardiff.ac.uk

Omer Rana
School of Computer Science and Informatics
Cardiff University, UK
ranaof@cardiff.ac.uk

Andrea Gaglione
Digital Catapult
London, United Kingdom
andrea.gaglione@digicatapult.org.uk

Alex Gluhak
Digital Catapult
London, United Kingdom
alex.gluhak@digicatapult.org.uk

Charith Perera
School of Computer Science and Informatics
Cardiff University, UK
pererac@cardiff.ac.uk

January 27, 2023

ABSTRACT

Cities are increasingly getting augmented with sensors through public, private, and academic sector initiatives. Most of the time, these sensors are deployed by having a primary purpose (objective) in mind (e.g., deploy sensors to understand noise pollution) by a sensor owner (i.e., the organization that invests in sensing hardware, for example, a city council). Over the last few years, communities undertaking smart city development projects have understood the importance of making the sensor data available for a wider community – beyond their primary usage. Different business models have been proposed to achieve this, including creating data marketplaces. The vision is to encourage new start-ups and small and medium-scale businesses to create novel products and services using sensor data to generate additional economic value. Currently, data are sold as pre-defined independent datasets (e.g., noise level and car park status data may be sold separately). This approach creates several challenges, such as (i) difficulties in pricing, which leads to higher prices (per dataset), (ii) higher network communication and bandwidth requirement, and (iii) information overload for data consumers (i.e., those who purchase data). We investigate the benefit of semantic representation and its reasoning capabilities towards creating a business model that offers data on-demand within smart city Internet of Things (IoT) data marketplaces. The objective is to help data consumers (i.e., small and medium enterprises (SMEs)) to acquire the most relevant data they need. We demonstrate the utility of our approach by integrating it into a real-world IoT data marketplace (developed by *synchronicity-iot.eu* project). We discuss design decisions and their consequences (i.e., trade-offs) on the choice and selection of datasets. Subsequently, we present a series of data modelling principles and recommendations for implementing IoT data marketplaces.

***K*eywords** Internet of Things, Semantic Interoperability, Data Discovery, Multi-dimensional Querying, Linked Data, Knowledge Management

1 Introduction

Over the last decade, many cities have initiated projects that deploy different sensors for various reasons. One popular application domain is air quality and environmental monitoring. After accomplishing the primary objectives, such data are often discarded (or stored somewhere where access can be difficult outside the initial project). There is often no mechanism (or motivation) to share data with outside parties (other than the organisation that deploy the sensing infrastructure). This approach leads to a waste of resources and can limit potential benefits derived from such data. Internet of Things (IoT) data marketplaces for smart cities are being proposed as a solution to address this challenge. *Urban data Exchange* [1] is one such data marketplace aimed at facilitating businesses to develop IoT- and AI-enabled services to improve citizens' lives and grow local economies.

Data marketplaces have received limited attention in the academic community. However, the buying and selling of data have taken place for a long time, especially within the business-to-business (B2B) context. Initially, these data transactions took place offline between companies and their alliances. Data have been widely sold across various domains, such as travel, advertising and insurance.

Even though we are not focusing on personal data marketplaces in this paper, we will briefly introduce and discuss them in the Related Work section. Typically a data marketplace comprises three types of stakeholders: (i) data buyers (who can also combine multiple data sources), (ii) data consumers and (iii) data brokers. The key contributions of this work are as follows:

- We propose an ontology aggregating other well-known ontologies to model sensor data in IoT marketplaces. We made design decisions during the ontology engineering process that led to different trade-offs. We discuss the trade-offs of each decision and highlight good practices observed in existing efforts.
- We propose a unique on-demand data offer creation technique. Buyers can create their custom data requests (i.e., data order) by considering four aspects: location, data type, date/time, and service level agreement.
- Through a series of use cases, we demonstrate the utility of knowledge engineering (including reasoning/ inferencing) in the context of data marketplaces. We also demonstrate different levels (data set level, market level, buyer level) of knowledge engineering approaches and their utility and related costs (e.g., computational complexity).
- We evaluate the performance of the proposed approach in three different distributed data marketplace setups. We measured some parameters and extracted several recommendations for future marketplace deployments.

2 Motivation and the Problem Definition

Currently, IoT data marketplaces sell data per entire dataset, as shown in Figure 1. For example, potential buyers could buy weather forecast data and parking status data in bulk. Each of the datasets may contain multiple pieces of data packed together in a pre-defined manner (e.g., `temperature`, `relativeHumidity` may be included in the Weather Forecast data offer). There are multiple problems with this approach.

Problem 1: Higher network communication and bandwidth requirement: ($\uparrow D_i^p \propto D_i^b \uparrow$) In the current approach, data offers are sold as predefined data bundles. There is no way to limit the number of data within a bundle that a buyer acquires (whether the buyer may request past (archival) data or future data (as a subscription)). The consequence of this approach is higher network download time and cost. For example, if a data consumer wants bicycle docking station data in Santander, they will need to buy the entire `Docking Stations - SAN` data set whether they need the entire data set or not. Therefore, there is a positive correlation between network communication, bandwidth requirement and volume of data. Given price of a data offer i is D_i^p and bandwidth required is D_i^b, price is proportional to bandwidth.

Problem 2: Difficulties in pricing which leads to higher prices: ($\uparrow D_i^p \propto D_i^v \uparrow$) Currently, each data bundle comprises large volumes of data. The cost of acquisition for a large volume of data is high. Therefore, the cost of the bundle has to be high as well. In a data marketplace, the price of a data offer has to cover the cost of data acquisition plus a profit margin. Therefore, there is a positive correlation between data prices and the volume of data. Given price of a data offer i is D_i^p and volume is D_i^v, price is proportional to volume.

Problem 3: Information overload for data consumers: ($\uparrow D_i^v \propto D_i^{pp} \uparrow$) .
In data science projects, 80% of time and effort is often devoted towards preparing the data (i.e., acquiring, cleaning, transforming, etc.), and only 20% is used to do the actual analysis. Therefore, the larger the buyers' dataset, the more effort they need to prepare and filter the relevant data. Assume that a given data analysis task

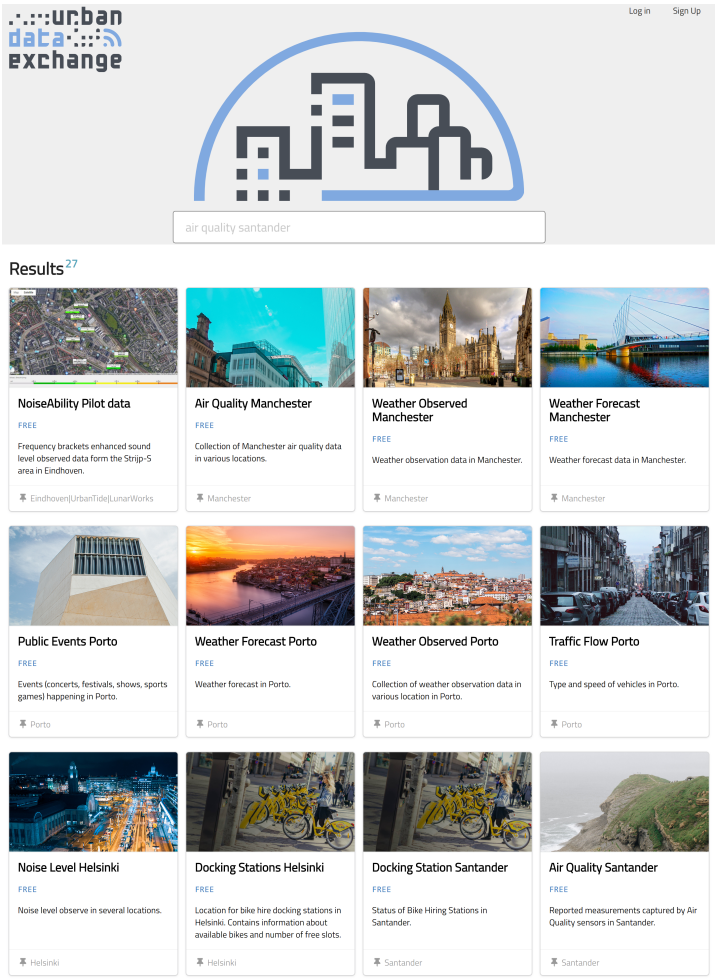


Figure 1: Urban Data Marketplace (Current Approach): FIWARE data models are used to organise the data into datasets. They have been harmonised to enable data portability for different applications, including Smart Cities, Smart Agrifood, Smart Environment, Smart Sensoring, Smart Energy, Smart Water, and others. The key weakness of this approach is that data buyers need to buy the entire dataset (e.g., Noise Level Helsinki) whether they need the entire dataset or not. This approach leads to higher data prices.

is related to `weekends` data (e.g., parking slot status). First, data scientists need to query the entire data set, remove the data related to `weekdays` and select only the data related to `weekends`. Therefore, the current bulk pre-defined data offering approach unnecessarily increases data scientists' workload (i.e., data consumers'). Given the size of a data offer i is D_i^v, the cost of data pre-processing is D_i^{pp} – cost of data pre-processing is proportional to volume.

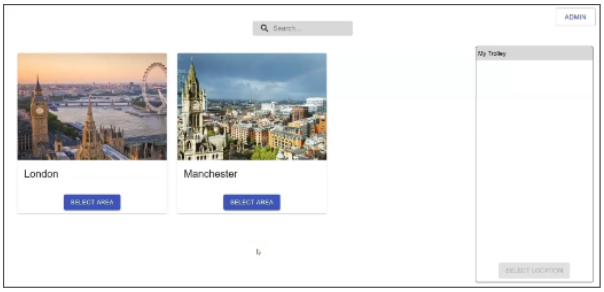
Problem 4: Limited data discovery capabilities: Currently, IoT data marketplaces organise datasets by type (broadly) and location. For example, it is usually up to the data seller to bundle the data into an offering as they see fit, as shown in Figure 1. Here, data offers are pre-defined and static without any mechanism to request customised data. Data search primarily relies on location. There is no way for data consumers to acquire `traffic data` in `London` on `rainy` days over the `last three years` in the current data marketplace scenario.

2.1 Design Principles and Architecture

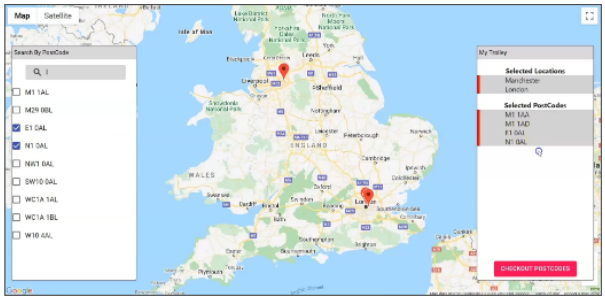
We suggest design principles to help data consumers communicate the data they need. These efforts yielded competency questions for our data model.

2.1.1 Design Principles

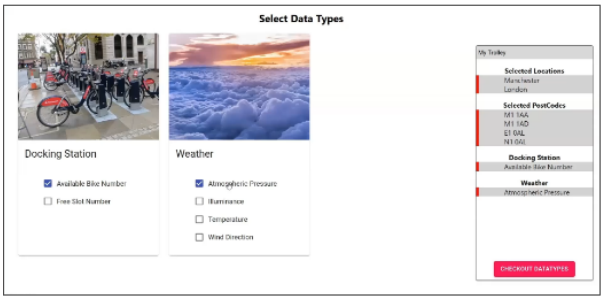
We interviewed ten data scientists to extract the following design principles. The identified expressed their priorities using questions-terms: (1) Where, (2) What, (3) When, and (4) How. Let us explain each of these design principles with a concrete example. As illustrated in a series of Figures (2(a), 2(b), 2(c), 2(d)), we have implemented the proposed IoT data marketplace by following these design principles.



(a) Starter Page of the Custom Data Request Builder



(b) Data consumers are provided with a UI to select the areas of interest



(c) Data consumers are provided with a UI to select the data types they need

(d) Data consumers are provided with a UI to express their Time requirements

Figure 2: Custom Data Request Builder

In order to guide potential data consumers, we have created a user interface as shown in Figure 2(a). Based on our preliminary investigation, the most critical decision data scientists make is to choose the area of interest (where). As illustrated in Figure 2(b), data consumers need to express the area of interest, and then available cities within the desired area will appear (e.g., London, Manchester). They can subsequently narrow down the search in a more granular way. For instance, they may select specific postcodes and zones or mark a particular area by drawing a polygon(s) on the provided map.

As illustrated in Figure 2(c), consumers can select which data types they want. Data are categorised into logical groups. We follow the data categories provided by FIWARE (`https://fiware-datamodels.readthedocs.io/en/latest/`) to organise the data types.

As illustrated in 2(d), the next stage allows data consumers to express their *'Time'* window of interest. First, data consumers need to scope their requests for the overall duration (e.g., last ten years, following four years). Secondly, data consumers are offered to narrow down their requests using an expressive set of semantic terms (e.g., weekdays, weekends, rainy days, sunny days, etc.)

These filtering terms is only possible due to semantic web technologies. We will discuss this capability (as well as the principles and technologies behind facilitating such capability) in detail later in this paper.

The next stage allows data consumers to express their Service Level Agreement (SLA) requirements, such as their *'co-location'* requirements. For example, a data consumer who has a focus on developing a *park and cycle* app. might want to understand `air` and `noise pollution`. In order for data to be meaningful, data consumers need to gather `air` and `noise pollution` data near `car parks` and `bicycle stations` (e.g., co-located within half a mile).

2.1.2 Architecture

The IoT data marketplaces need to be distributed in nature. Data owners are expected to store and manage data items and only share their metadata with brokers like IoT data marketplaces. Consequently, when a broker receives a request, it knows from where to gather the data (or to decide whether it is possible or not to fulfil a given request).

Figure 3 depicts the architecture of the IoT data marketplace and details are presented here (`https://gitlab.com/synchronicity-iot`). In summary, we have developed a user interface that allows data consumers to build their data requests. We then organise each data request using a standardised JSON schema and send it to the validation engine. The validation engine determines whether the IoT data marketplace can fulfil a given request based on the available metadata. Then, one or multiple SPARQL queries will be generated based on the requirements of the data request (and depending on where the actual data reside).

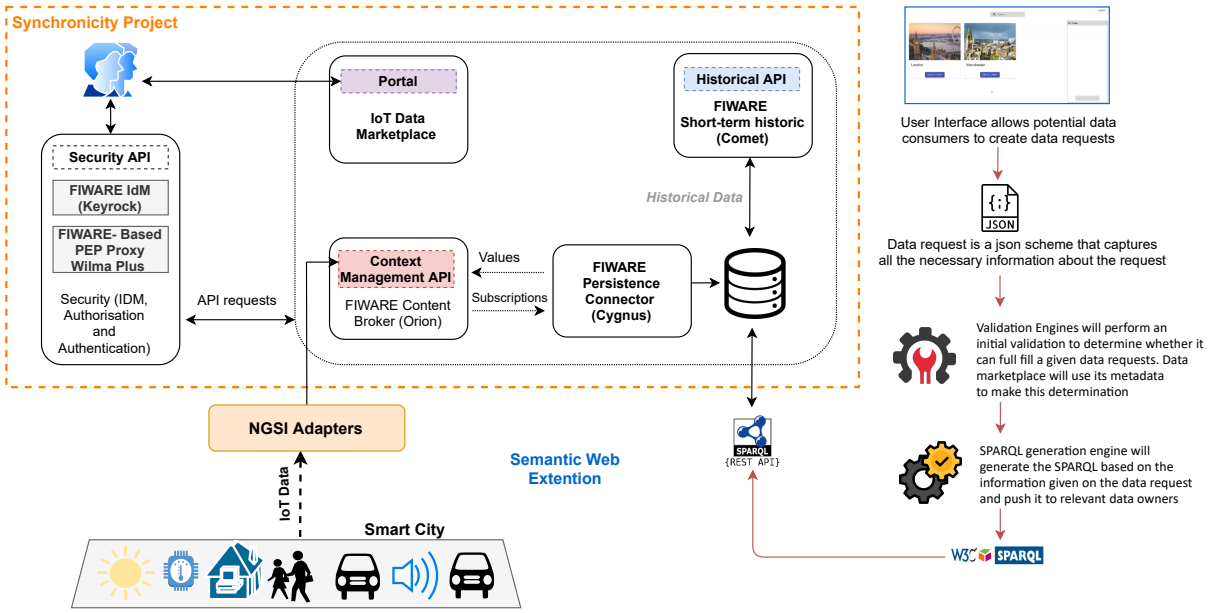


Figure 3: Semantically Enhanced IoT Data Marketplace Architecture

3 Data Marketplace Design

To address the requirements stated in 2.1.1, we propose an IoT data marketplace that allows potential data consumers to buy *only the data points/records they need to solve a given problem*. Pre-defining a large number of data offerings is not feasible; therefore, the best approach is to allow consumers to create their data offerings (i.e., data requests). To illustrate the notion, assume a new tourism company is interested in purchasing various data entities to build an AI decision support system. These entities may contain specific observations about local attractions such as beaches, museums, parking spots, and bike docking stations. Thus, the organisation prepares a single order that includes the needed data records from the datasets instead of acquiring the entire dataset for each entity separately. Nonetheless, this method has several drawbacks in terms of the following: (see table 1).

- **Data pricing** could be complicated because each data source may price its observations- depending on size and novelty. In addition, the broker fees and any variable extra charges have to be carefully calculated and added to the total bill.

- **Data publishing** could raise privacy and ownership concerns because data providers may have different privacy policies and credit preferences. Therefore, appropriately tailored privacy and data ownership agreements must exist to satisfy all parties.

Table 1: Possible Scenarios

Criteria	Current Pre-Defined Data Offering Approach	Proposed On-Demand Data Offering Approach
• Pricing Structure	Simple	Could be complicated
• Pricing Fairness	Less fair	More fair
• Data Discoverability	Less discoverable	More discoverable
• Publishing Complexity	Easy and simple to publish	Could be complicated to publish
• Data Preparation Complexity (from a data consumer perspective)	Higher (as large data sets need to be processed and filtered)	Less (as data is already processed and filtered)

3.1 Data Model

Our data model approach comprised a foundational ontology instantiated with six heterogeneous datasets. The ontology's purpose is to describe sensor data in the IoT marketplace. We followed the NeOn methodology [2, 3, 4] to develop the ontology. Although there are numerous other ontology development methodologies [5, 6, 7, 8, 9], we selected NeOn as it has multiple modular scenarios to choose from and adopt to our current requirement. Figure 4 shows the ontology development's life-cycle. We adopted the first, second and third scenarios. The first scenario outputs the Ontology Requirement Specification Document (ORSD) as shown in the Appendix A. Then, from the second scenario, we identified the non-functional requirements based on the ORSD (see Appendix). We followed the process of reusing existing ontological resources from the third scenario. Following that, we implemented and evaluated the ontology using various tools. Figure 5 depicts the proposed core ontology, Table 2 discusses the key characteristics of these ontology, and figure 6 shows the ontology instantiated with six sensor datasets.

3.1.1 Ontology Requirements

The first step for developing the ontology was to gather the required information. Here, we used the information collected at the design stage (see section 2.1.1- formulating competency questions to develop the ontology. During this step, we named our ontology the Urban Data Exchange Ontology (UDEO). We produced the ORSD (see Appendix A), which contains the conceptual building blocks for the ontology as follows:

- Ontology purpose: to describe the data sensors data.
- Ontology scope: The Internet of Things (IoT).
- Ontology implementation language: OWL.
- Ontology intended users: Small to medium businesses (SMEs).
- Ontology non-functional requirements: listing elements that must be included in the ontology, such as IoT geospatial and time classes.
- Ontology functional requirements: contain the Competency Questions (CQs) to build and validate the ontology.

3.1.2 Ontology Analysis

In this step, we revised the competency questions from the requirements phase and extracted knowledge to implement the ontology. We decided to model the following concepts: 1) sensor data observations, 2) sensing infrastructure, 3) location, 4) temporal aspects, and 5) units of data. Here, we drew the conceptual ontology diagram from which we followed to encode the digital ontology format. It is worth noting that we only used a subset of existing ontologies in UDEO, making it extendable for future work

Sensor data observations: To support the requirements, we first need an ontology to model sensor data observations (e.g., Temperature: 21C). One of the widely used ontologies is Semantic Sensor Network (SSN) ontology. SSN follows a horizontal and vertical modularization architecture by including a lightweight. However, self-contained core ontology called SOSA (Sensor, Observation, Sample, and Actuator) [10] for its elementary classes and properties. We mostly adopted SOSA concepts and properties. We acknowledge that many ontologies - as stated by Bajaj et al. [11]- are derived from or based on SSN, such as IoT-Lite [12], SAREF [13] and, Fiesta-Priv [14]. However, we adopted concepts directly from SSN ontology due to its wide usage.

Sensing infrastructure: SSN provides sufficient facilities to model sensing infrastructure. Therefore, we also reused the concepts and properties for this purpose.

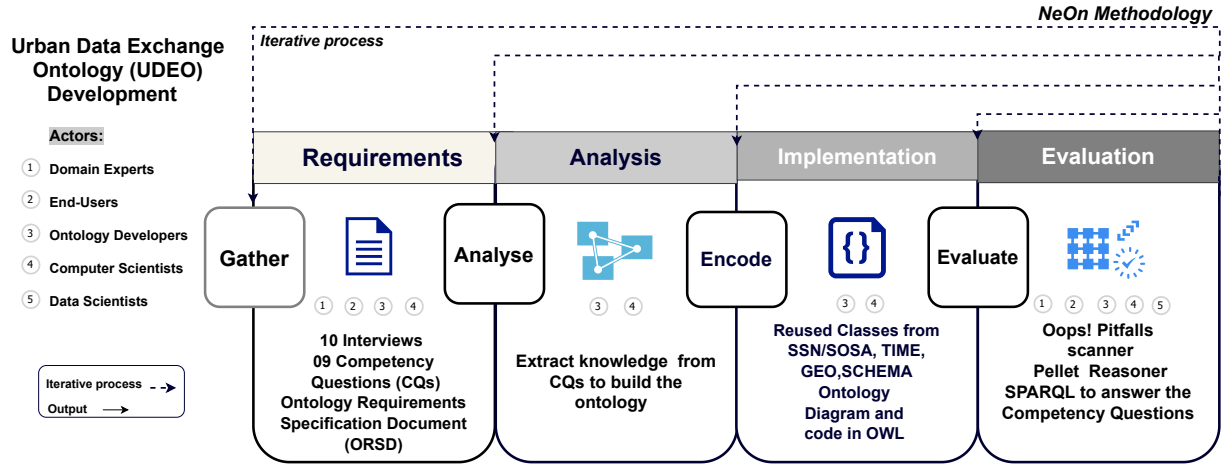


Figure 4: Neon Methodology for developing the proposed Urban Data Exchange Ontology

Location: This work focuses on outdoor locations, easily identified using GPS Coordinates.

Temporal aspects: We decided to store the timestamp of each observation using XML DateTime data type (i.e., xsd:dateTime). We considered OWL-Time [15] to be incorporated into our ontology. However, we concluded that SPARQL's Time function also provides most of the OWL-Time capabilities. Therefore, we omitted it from our ontology.

Units of data: We adopted the concepts from Quantities, Units, Dimensions, and Types Ontology (QUDT) Ontology to model data units. It is the most used ontology for this purpose and provides most of the necessary units for our application.

3.1.3 Ontology Implementation

As mentioned in the analysis step, we mainly adopted most of the UDEO classes from SSN/SOSA ontology [10]. The conceptual ontology model, or the lightweight version, was designed as a UML diagram in draw.io software. The ontology diagram in 6 was discussed between UDEO stakeholders before its digitization. Each of the concepts and relationships modelled in UDEO is listed in Table 2. We encoded the digital version in the Protege ontology editor and exported it to a dedicated knowledge graph platform [16].

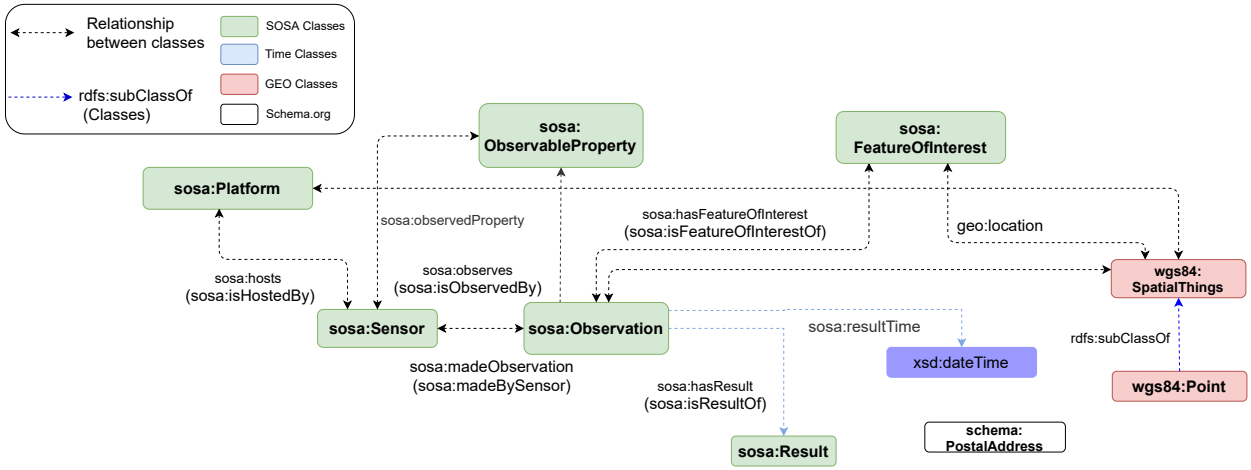


Figure 5: Proposed Ontology for Internet of Things (IoT) Data Marketplace

Table 2: Proposed data model: Concepts and relationships. [Official definitions are in *Italics*]

Concept	Description
Observation (OWL Class) [sosa]	*Act of carrying out an (Observation) Procedure to estimate or calculate a value of a property of a FeatureOfInterest (e.g., Room).* Observation can be seen as a placeholder that links relevant information together. As illustrated in Figure 6, observation can be considered an ID for each data record in our data model. Each raw depicts a data record.
ObservableProperty (OWL Class) [sosa]	*An observable quality (property, characteristic) of a FeatureOfInterest.* (e.g., Temperature, humidity, presence)
Sensor (OWL Class) [sosa]	*Device, agent (including humans), or software (simulation) involved in, or implementing, a Procedure.* (e.g., Temperature sensor, humidity sensor, motion sensor). In our model, we have created a unique ID for each sensor based on its hosted platform.
Platform (OWL Class) [sosa]	*A Platform is an entity that hosts other entities, particularly Sensors, Actuators, Samplers, and other Platforms.* In UDEO, sensors are attached to different types of platforms, as shown in Figure6. We do not necessarily keep track of the exact location of the platform. However, location can be approximately identified by using the feature of interest.
FeatureOfInterest (OWL Class) [sosa]	*The thing whose property is being estimated or calculated in the course of an Observation to arrive at a Result, or whose property is being manipulated by an Actuator, or which is being sampled or transformed in the act of Sampling.* In the context of UDEO, *BuidlingSpaces* are the *FeatureOfInterest* (e.g., offices, zones, floors). Most of the sensors are used to observe a property (phenomenon) of a location (e.g., the temperature in a room) [17].
Result (OWL Class) [sosa]	*The Result of an Observation, Actuation, or act of Sampling. To store an observation's simple result value one can use the hasSimpleResult property.* Result is a placeholder to link related information, such as values and units. The UDEO model stores the data value and its unit type.
resultTime (Datatype Property) [sosa]	*The result time is the instant of time when the Observation, Actuation or Sampling activity was completed.* Each data record in the UDEO system comes with a time stamp. We attach a timestamp to each observation using this data property.
SpatialThing (OWL Class) [WGS84]	*A class for representing anything with a spatial extent, i.e., size, shape or position.*

3.1.4 Ontology Evaluation

This step aims at evaluating the ontology quality in-terms of *structure, semantic representation and interoperability*. To evaluate the structure and semantic representation, we used the Open-source online scanner, Oops![18] and Pellet reasoner[19] built-in Protege editor. Following that, we fired SPARQL queries towards the linked data (i.e., UDEO instantiated with IoT datasets) to answer the competency questions (CQs) listed in the Appendix.

3.2 Experimentation Plan

Our experimentation plan consists of three layers, as shown in Figure 17, (i) data sources, (ii) adaptor layer (iii) evaluation. Our objective is to simulate a data marketplace using the most practical solution that fits the purpose.

- Data Sources: We expanded the UDEO to accommodate six data sources: docking stations, air quality, noise level, parking status, museums, and beaches. Further, we generated synthetic datasets for each data source that adhere to the FIWARE data models structure.
- Adaptor Layer: We mapped the six datasets into the Resource Description Framework (RDF) graph - referencing UDEO.
- Evaluation Layer: We stored the data from the adaptive layer in triple-store databases under three different architectures and evaluated each one's performance.

SynchroniCity IoT Data Marketplace receives data as JSON files (modelled using FIWARE standards) from data owners/publishers. SynchroniCity IoT Data Marketplace currently has a limited amount of data. It was not sufficient for us to conduct an experiment to uncover the utility of semantic web technologies and their impact on computing infrastructure. Therefore, we developed an algorithm to produce the required number of synthetic sensor data observations in JSON format (depending on the specific experiment). Then we transformed JSON data into Resource Description Framework (RDF) graph and loaded graph data into a triplestore database. We meant that our algorithm could run and update either concurrently or partially by modular. For example, the user may run the algorithm's first part to generate JSON data and then transform the output JSON file into RDF using the second part as an independent code. Our experience found that running the algorithm by stages consumes less time when generating many data

observations (e.g., 1000K+). Code Snippet 1 explains the technique used. The UDEO and datasets were connected to create fully-fledged data models. We employed Stardog [16], a knowledge graph platform that integrates heterogeneous and isolated data sources. Stardog hosts the triplestore databases and has an IDE (Stardog studio) capable of performing numerous operations, SPARQL, GraphQL, artificial intelligence and machine learning. We wrote complex SPARQL queries to test the efficiency of retrieving information and inferring new phenomena. Further details are in the evaluation section.

Code Snippet 1: Data Generate, Transform and Load

```
Part 1– Generate
Function GenerateModel():
    Function GenerateId(size, chars + string):
        return join (chars for i in range(size))
        model = DataModel(Id ,type)
        n = name
        v = value
        model.add(n,v)
    return model
x = observation number (integer only)
for i in range(x) do
    data= GenerateModel()
    JSON.dump(data)
end for
Part 2 –Transform
Data = ReadJSON(data)
for i, r in Data: do
    RDFData=Graph.add(subject,property,object)
    Graph.serialise(RDFData,format=turtle)
end for
Part 3 – Load
StardogConnection=(endpoint,username,password)
DatabaseName = NewDatabase(RDFData)
connection = ConnectStardog(RDFData, StardogConnection)
connection.add(RDFData, format=turtle)
connection.commit
```

4 Evaluation

To demonstrate the proposed approach's utility and feasibility, we evaluated it using three different architectures where each differs in how it stores and queries data. In the first instance, we used Code Snippet 1 to generate three incremental series of synthetic datasets. The number of generated observations was equal for each one. However, the volume varied when serialised into RDF graphs. Table 3 shows the Kilobyte (KB) size for the generated RDF graphs. Furthermore, Figure 8 compares the volume of each data model. It was evident that the Noise Level dataset with more than 1K triples is the largest. That is due to the increased amount of metrics (e.g. CO, NOx) in a single observation compared to other data sources. The objective is to estimate how much data can be stored in a disk for each data model. It helps to understand how much data storage is required for a given use case, depending on the frequency of the observations captured and the number of metrics modelled within each observation. We then interrogated the data stores to answer competency questions such as *where can I park and ride?*. We developed complex SPARQL queries and executed them across the databases in question. Then, we inserted Semantic Web Rule Language (SWRL) into the databases and reran the queries. We assessed each approach by critically analysing the query response time on databases with and without SWRL. Subsequently, we reflected on the impact of complex queries and reasoning processes on data- highlighting some strengths and weaknesses. Furthermore, we analysed the collected data to compare the results and determine if response time is faster after inserting SWRL rules. In other words, to examine if reasoning and setup reduce the query response time. Finally, key findings suggested the most suitable approach for the data marketplace. The accumulated response time observations were analysed with the following steps to detect the difference between the two groups:

- Line graph to visualise and compare the data.
- Testing data distribution with Shapiro-Wilk normality test to determine the appropriate statistical test for detecting the difference between two groups (i.e., parametric or non-parametric test). Here, the histograms and

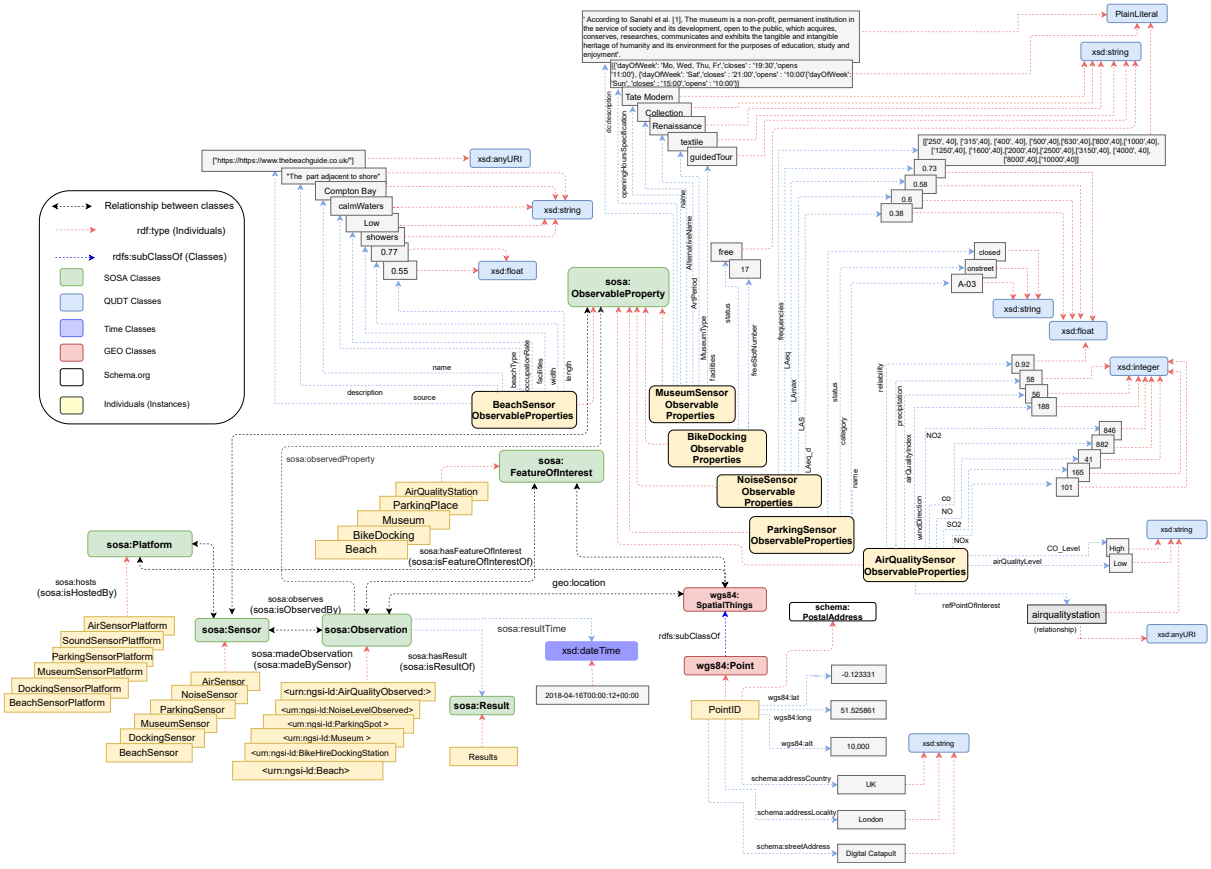


Figure 6: UDEO Ontology instantiated with six sensors' datasets

test p-value suggested that the observations were not found to be not normally distributed. Accordingly, we conducted non-parametric tests.

- Using Kurskal-Wallis to check if the two groups of observations (No-rule-SWRL) are related (i.e., sampled from the same distribution)
- Finally, Mann-Whitney, the statistical test proved if there is a significant difference between the two groups. In other words, to examine if SWRL increased or reduced the query efficiency and response time.

Therefore, the claimed hypotheses can be formulated as:

Hypothesis H_0: SWRL reduces query response time.

Hypothesis H_1: SWRL does not reduce query response time.

The significance level of

$$\alpha = 0.05$$

If the p-value is greater than the significance level, we retain the null hypothesis.

4.1 Rule-based Reasoning

In the context of the semantic web in general and RDF graphs in particular, reasoning also referred to as inferencing, derives a new phenomenon from a given dataset- based on named axioms, applicable rules and definitions in the data model. Reasoning rules are declarative and represent proven knowledge or concepts modelled by experts. Rule-based inferring uses conditional IF-Then entailment rules. The logical consequences in the IF clause are inferred in the statement of Then. For example, we may deduce outdoor activities are busier than usual on sunny days. Reasoning can reshape and align data, creating new views of data and connections. More importantly, it validates domain modelling and detects violations. One of the features is the reasoning at query time. Besides the excellent performance, it allows users to specify the type and pay only for its reasoning usage. Reasoning can be enabled or disabled via a simple

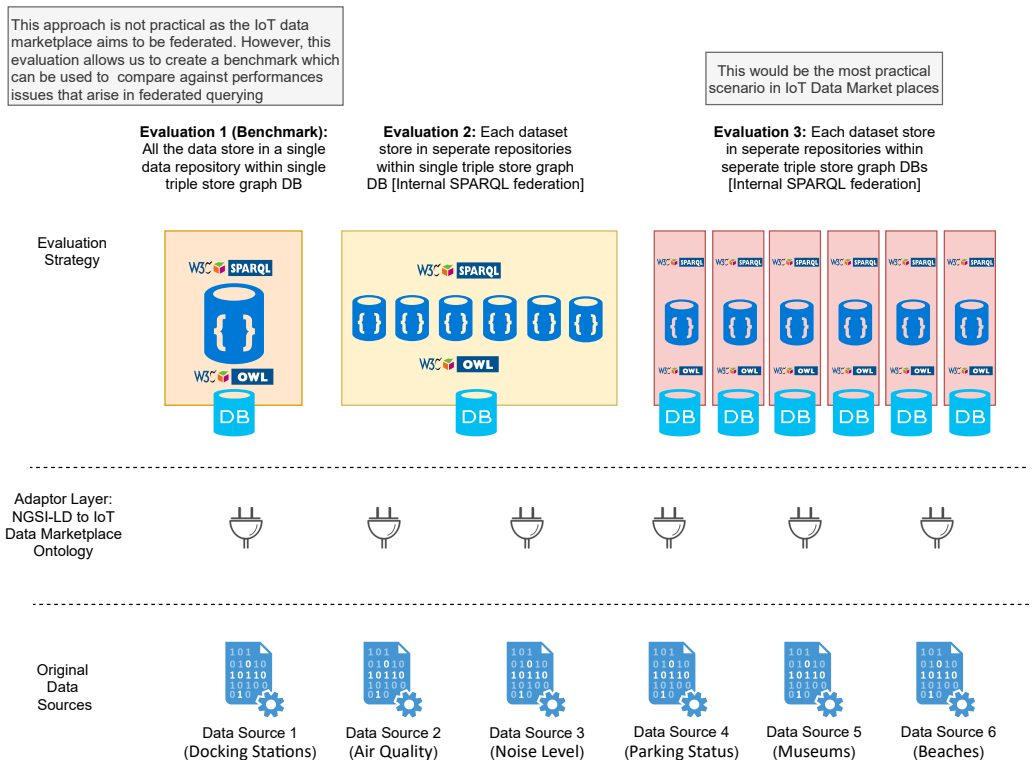


Figure 7: Experimental Setup

Data Model	1K	10K	100K
Air Quality	90	903	9023
Beaches	108	859	8580
Docking Stations	119	1188	11877
Museums	153	1215	12141
Noise Level	135	1347	13471
Parking Status	99	918	9180

Table 3: Data Model RDF Graph Size (KB)

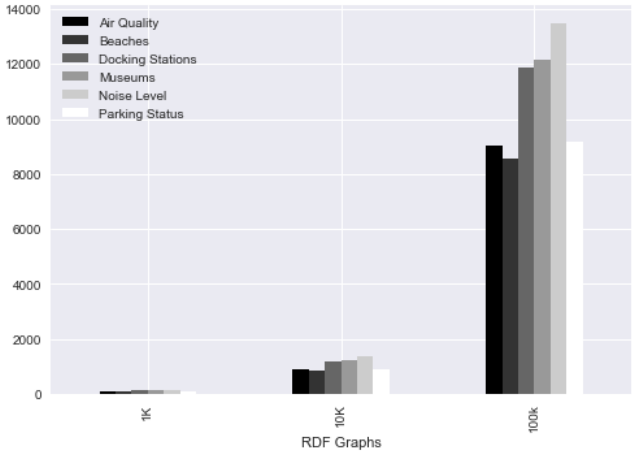


Figure 8: Comparison between Data Model Sizes(KB)

boolean command. When enabled, rules or axioms are triggered, and reasoning executes according to its value in the database. In this case study, we created a rule that assumes low rental bike availability during sunny days. We sketched this assumption to prove a concept that may not reflect objective reality. As discussed in the coming sections, the sunny days' rule is injected into our database and used to evaluate its performance in three different scenarios.

Code Snippet 2: SunnyDays Rule

```
prefix rule: <tag:stardog:api:rule:>
[] a rule:SPARQLRule ;
rule:content """
PREFIX fiware:<https://uri.fiware.org/ns/data-models#
IF
{?id a fiware:BikeHireDockingStation;
    fiware:AvailableBikeNumber ?AvialableBikeNumber;
BIND(xsd:integer(AvialableBikeNumber) <5  AS ?SunnyDays)}
THEN
{?id  fiware:SunnyDays  ?SunnyDays}""" .
```

Code Snippet 3: PREFIXES for *"Where can I park my car and ride a bicycle?"*

```
PREFIX fiware:<https://uri.fiware.org/ns/data-models#>
PREFIX ngsi: <https://uri.etsi.org/ngsi-ld/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <https://schema.org/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geof:<http://www.opengis.net/def/function/geosparql>
PREFIX unit: <http://qudt.org/vocab/unit#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

4.2 Evaluation One

The first experiment loaded all six datasets and our ontology (UDEO) in a single database. We executed the SPARQL query scripts at Code Snippet 4 and 5 hundred times consecutively and respectively. The queries yield vacant car parking spots and available bikes near a geographical location. We recorded the time taken by each script in milliseconds (ms). We repeated the experiment after inserting SWRL, which could infer a sunny day. The SPARQL queries were executed 100 times before and after reasoning (i.e., inserting SWRL). Figure 9 shows the query responses for each query type (i.e., No-rule and SWRl) and the histograms in 12 unveil the data distribution nature. The obtained p-values from Shapiro-Wilk, Kruskal-Wallis and Mann Whitney, as listed in table 24, were far below the significance level of 0.05. Therefore, we reject the null hypothesis and conclude that SWRL does not reduce response time in this query setup.

Code Snippet 4: Evaluation One NoRule Result Response Time = 237 ms

```
Select *
{?id a  fiware:ParkingSpot;
        fiware:category ?category;
        fiware:dataProvider ?dataProvider;
        ngsi:status ?status;
        ngsi:location ?location;
        ngsi:ParkingPoint ?ParkingPoint.
?id2 a  fiware:BikeHireDockingStation;
        fiware:availableBikeNumber ?availableBikeNumber;
        schema:address ?address;
        ngsi:status ?Bikestatus.
sosa:PointID a pos:Point;
           pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance
    (?ParkingPoint, ?SOSAPoint, unit:Kilometer)
    as ?Distance).
FILTER(xsd:integer(?Distance < 500))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}}
LIMIT 1
```

Code Snippet 5: Evaluation One-SWRL-Result Response Time= 571 ms

```
Select *
{?id a  fiware:ParkingSpot;
        fiware:category ?category;
        fiware:dataProvider ?dataProvider;
        ngsi:status ?status;
        ngsi:location ?location;
        ngsi:ParkingPoint ?ParkingPoint.
?id2 a  fiware:BikeHireDockingStation;
        fiware:availableBikeNumber ?availableBikeNumber;
        schema:address ?address;
        ngsi:status ?Bikestatus;
        fiware:SunnyDays ?SunnyDays.
sosa:PointID a pos:Point;
           pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance
    (?ParkingPoint, ?SOSAPoint, unit:Kilometer)
    as ?Distance).
FILTER(xsd:integer(?Distance < 500))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}}
LIMIT 1
```

4.3 Evaluation Two

The second experiment stored each dataset with the UDEO locally but in separate databases using the same Stardog instance. Here, the SPARQL federation answered the competency question. For instance, the query targeted the data source internally by using the SERVICE keyword with the URI typed as *db://database* instead of the SPARQL endpoint URL. We executed the SPARQL queries shown in Code Snippet 6 and 7 hundred times, with and without SWRL-recording observations, for further analysis. In the same manner, the line plots 13 compared the two variables, while the histograms in 15 suggest that the observations do not follow the normal distribution. Noticeably, this setup (locally separated databases) has a quicker response time than the unified database in evaluation one. Similar to evaluation one 4.2, the p-values from Shapiro-Wilk, Kruskal-Wallis and Mann-Whitney, as listed in table 24, were less than the significance level of 0.05. Once again, we reject the null hypothesis and recommend injecting SWRL into the internally distributed databases does not accelerate response time.

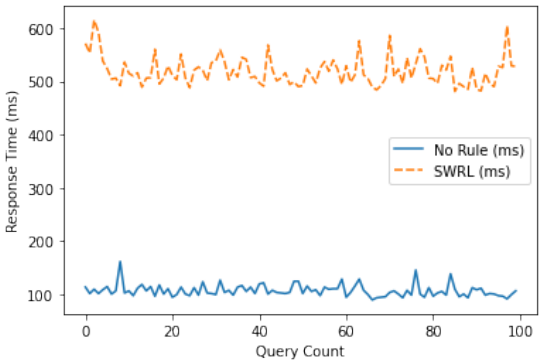


Figure 9: Evaluation 1 visualisation

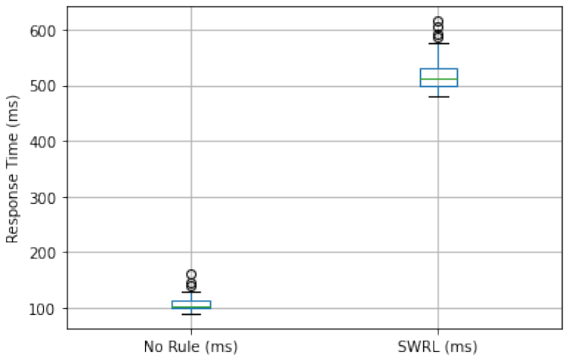


Figure 10: Evaluation 1 boxplot

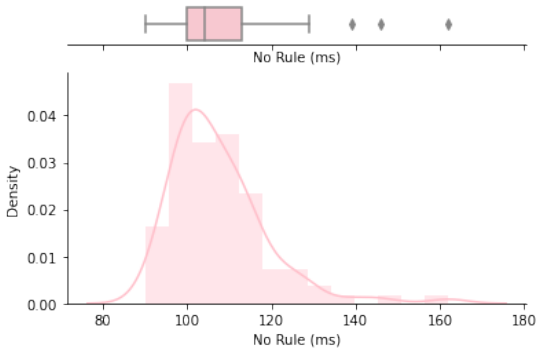


Figure 11: Evaluation 1 No-Rule distribution plot

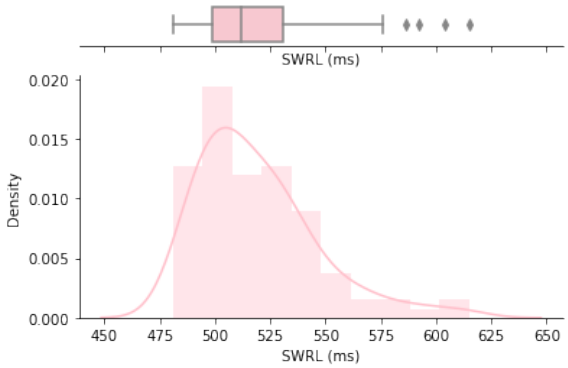


Figure 12: Evaluation 1 SWRL distribution plot

Code Snippet 6: Evaluation Two NoRule Result Response Time = 32 ms

```
SELECT *
{SERVICE <db://BikeHireDockingStation100k>
    {?id a  fiware:BikeHireDockingStation;
            fiware:availableBikeNumber ?availableBikeNumber;
            schema:address ?address;
            ngsi:status ?Bikestatus.}
{SERVICE <db://Parking>
    {?id2 a fiware:ParkingSpot;
            fiware:category ?category;
            fiware:dataProvider ?dataProvider;
            ngsi:status ?status;
            ngsi:location ?location;
            ngsi:ParkingPoint ?ParkingPoint.
sosa:PointID a pos:Point;
pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance
     (?SOSAPoint,?ParkingPoint, unit:Kilometer)
    as ?Distance).
FILTER(xsd:integer(?Distance < 290))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}}}
LIMIT 1
```

Code Snippet 7: Evaluation Two-SWRL-Result Response Time= 47 ms

```
SELECT *
{SERVICE <db://Parking>
   {?id2 a  fiware:ParkingSpot;
            fiware:category ?category;
            fiware:dataProvider ?dataProvider;
            ngsi:status ?status;
            ngsi:location ?location.}
            # ngsi:ParkingPoint ?ParkingPoint.
    {?id a fiware:BikeHireDockingStation;
            fiware:availableBikeNumber ?availableBikeNumber;
            fiware:AvailableBikeNumber ?AvialableBikeNumbe;
            schema:address ?address;
            ngsi:status ?Bikestatus;
            fiware:SunnyDays  ?SunnyDays.
sosa:PointID a pos:Point;
pos:SOSAPoint  ?SOSAPoint.
# BIND (geof:distance
    #?SOSAPoint, ?ParkingPoint, unit:Kilometer)
    #as ?Distance).}
# FILTER(xsd:integer(?Distance < 290))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))}}
LIMIT 1
```

4.4 Evaluation Three

In the last experiment, we stored each database in a separate computer node under different Stardog instances. Every machine acted as an independent data provider. We executed federated SPARQL queries illustrated in Code Snippet 8 and 9 to answer our competency question from the desired database without moving or copying data. This time, the query targeted a SPARQL endpoint in a remote machine. Therefore, an IP address was required along with the port number to reference with SERVICE Keyword. Nevertheless, HTTP authentication was also necessary to access the reference SPARQL endpoint. It can be achieved by disabling Stardog security on startup or storing password credentials in the Stardog directory. The same query for all other evaluations was executed concurrently, in the same manner, with

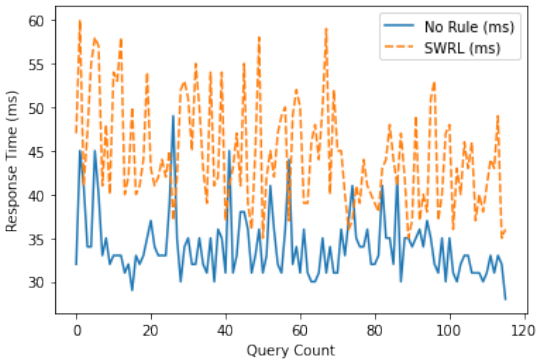


Figure 13: Evaluation 2 visualisation

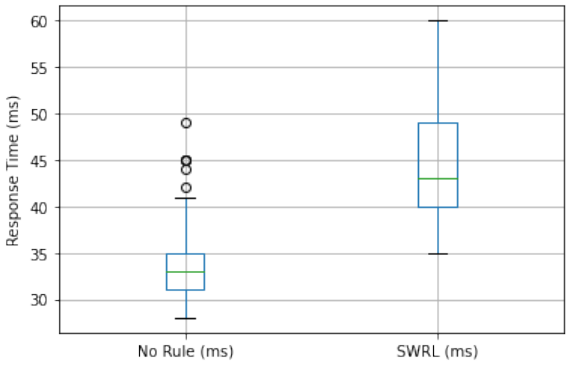


Figure 14: Evaluation 2 boxplot

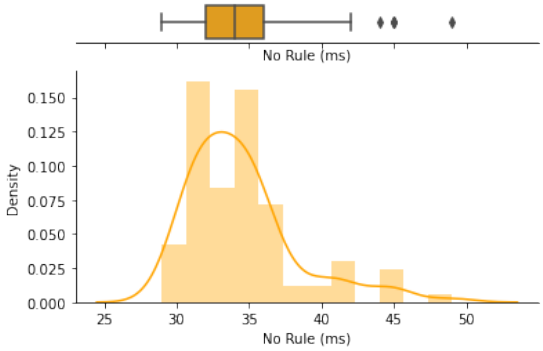


Figure 15: Evaluation 2 No-Rule distribution plot

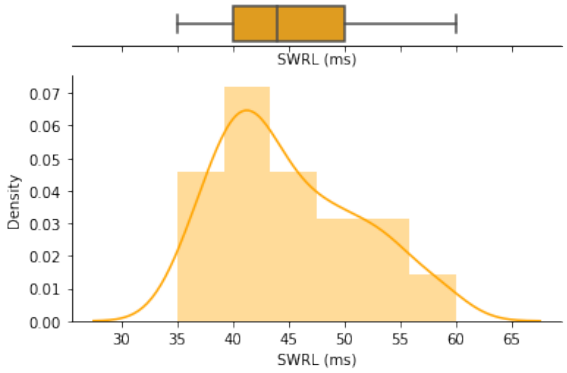


Figure 16: Evaluation 2 SWRL distribution plot

and without SWRL. Thus, response times were recorded for analysis (i.e., table 24) and comparison (i.e., table 18) purposes. Astonishingly, the average response time with SWRL was faster than no-rule. Accordingly, we retain the null hypothesis and conclude that SWRL reduces the query response time in the decentralised data storage manner.

Code Snippet 8: Evaluation Three NoRule Result Response Time = 99 ms

```
SELECT *
 {SERVICE <http://192.168.0.128:5820/Parking/query>

     {?id2 a fiware:ParkingSpot;
           fiware:category ?category;
           fiware:dataProvider ?dataProvider;
           ngsi:status ?status;
           ngsi:location ?location.}
{SERVICE <http://192.168.0.128:5820/Bike/query>

    {?id a  fiware:BikeHireDockingStation;
           fiware:availableBikeNumber ?availableBikeNumber;
           fiware:AvailableBikeNumber ?AvialableBikeNumber;
           schema:address ?address;
           ngsi:status ?Bikestatus;}}
LIMIT 1
```

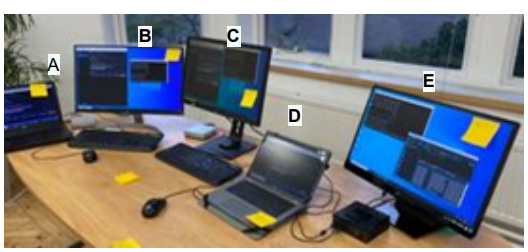
Code Snippet 9: Evaluation Three SWRL Result Response Time = 86 ms

```
SELECT *
{SERVICE <http://192.168.0.128:5820/Parking/query>
{?id2 a fiware:ParkingSpot;
       fiware:category ?category;
       fiware:dataProvider ?dataProvider;
       ngsi:status ?status;
       ngsi:location ?location.}

{?id a  fiware:BikeHireDockingStation;
       fiware:availableBikeNumber ?availableBikeNumber;
       fiware:AvailableBikeNumber ?AvialableBikeNumber;
       schema:address ?address;
       ngsi:status ?Bikestatus;
       fiware:SunnyDays  ?SunnyDays.}}
LIMIT 1
```

4.5 Results

Data aggregated (n=100) for each evaluation was fed to a python code for visualisation, exploration and statistical testing. Figure 22 are line graphs that visualise the flow of each trail. Querying the database after inserting SWRL took more time than querying than database without a rule. We performed a descriptive analysis to understand the data. Then, Shapiro Wilk, Kruskal-Wallis, and Mann Whitney statistical tests to check the distribution of the samples and detect and compare differences between the two independent samples (no-rule/SWRL) for each experiment, respectively. *Descriptive Analysis* in Figures 23 explored the datasets to help in understanding the data content and characteristics. For example, the line graphs for evaluation one indicated a considerable difference between query time responses for



Name	Data Provider	CPU	SSD (GB)	RAM (GB)
A	Laptop 1	Intel® Core™ i7-6Gen@2.60GHz(8CPUs)	480	16
B	MiniPc3	Quad-Core Intel Atom Z8350 @1.44 GHz	128	8
C	MiniPc2	Quad-Core Intel Atom Z8350 @1.44 GHz	128	8
D	Laptop2	Intel Core i7 1.8GHz 10Gen	512	16
E	MiniPc1	Intel® Core™ i3 5Gen@2.00GHz (4CPUs)	60	8

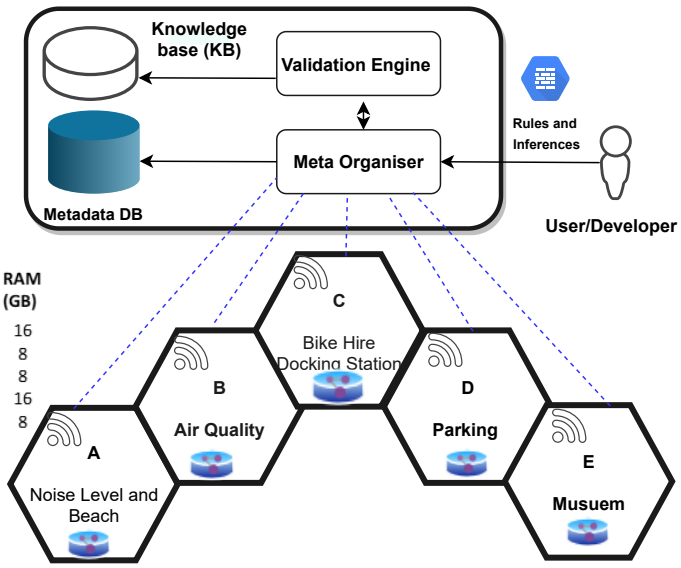


Figure 17: Evaluation Three Setup, A to E represent the different data providers' machines. Data providers may store their data on edge using mini pcs or dedicated computers. Here, the user/developer places a request. It gets verified by the validation engine and passed to the meta organizer that knows the data provider that has the requested information.

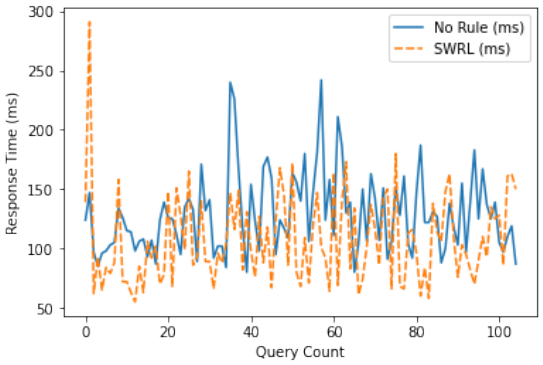


Figure 18: Evaluation 3 visualisation

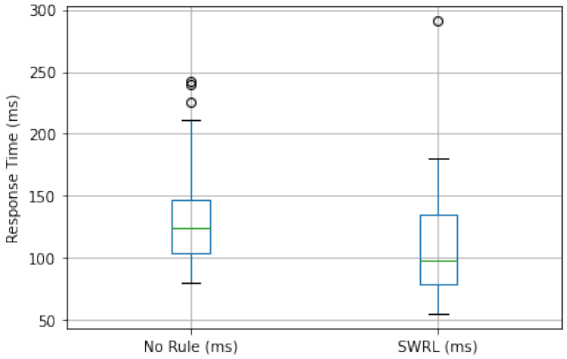


Figure 19: Evaluation 3 boxplot

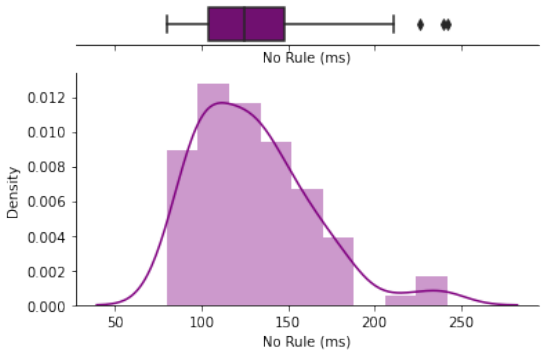


Figure 20: Evaluation 3 No-Rule distribution plot

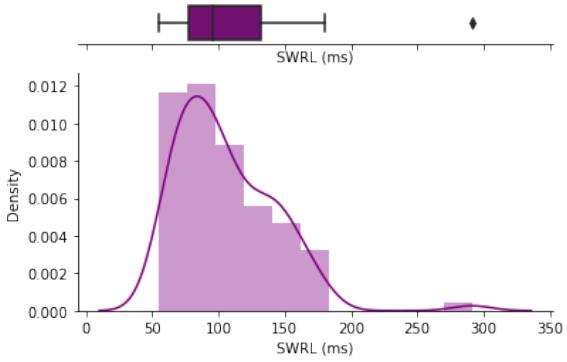
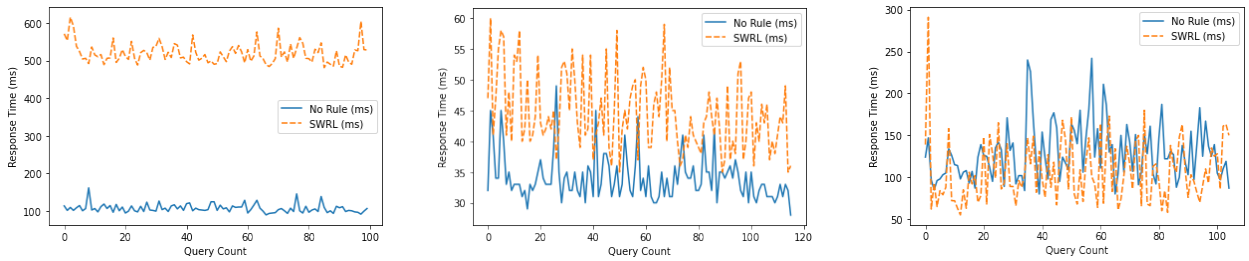


Figure 21: Evaluation 3 SWRL distribution plot

the dataset with and without SWRL. Unlike evaluation two, where the gap narrowed dramatically, it was surprising that querying with SWRL was slightly faster in evaluation 3. Presumably, this result could be due to the distribution of the datasets over disparate nodes and the remote access, which distinguished evaluation three from the others. The different mini pcs nodes that hosted the datasets had relatively small disk space ranging between 32GB to 128G and no less than



(a) (b) (c)

Figure 22: Compare the three evaluations query response time (ms), with and without reasoning rules inserted. (a) Evaluation One (All datasets and the ontology stored in a single database). (b) Evaluation Two (Each dataset and ontology is stored locally in separate databases). (c) Evaluation Three (Each dataset and ontology stored remotely with autonomous data providers)

8GB of RAM. Noticeably, the number of observations was equal in the three experiments, the mean fluctuated between approximately 34.45 and 519.13, and the standard deviation was at its minimal of 3.78 in evaluation two. To further discover the data distribution, the histograms and *Shapiro-Wilk* served the purpose of deciding the most appropriate statistical tests. It was clear from the histograms that the data in the three evaluation datasets do not resemble bell curves. Shapiro-Wilk test confirmed the non-normality further through the p-values (<0.05). Therefore, we rejected H0 with a 95 per cent confidence interval and concluded that all datasets do not follow the normal distribution. *Kruskal-Wallis test* was the selected non-parametric test that suited our data distribution. It suggested that the two samples (no-rule/SWRL) for each experiment came from different distributions with p-values of less than 0.05. Figure 24 explained in detail the hypotheses result of the evaluations based on their p-values, the final test, *Mann-Whitney*, was used to determine if the response time was different after inserting SWRL. The result also suggested a significant difference between the samples of each experiment (no-rule/SWRL).

4.6 The Distributed IoT data marketplace with reasoning Approach

Distributed semantic data sources with reasoning (i.e., SWRL) are proven feasible in the IoT data marketplace. It provides dynamic access to various moving or copying them. Thus, data owners keep and maintain their data and only share their metadata with the IoT data marketplace. The results show that semantic data sources use the network very efficiently by transferring small-size packets. The query response time is faster when reasoning is applied. Before system deployment, the information overhead and implementation cost must be considered.

- Information overhead can typically originate from the data structure, runtime, and data exchange. Here, the semantic data sources are database engines in which data are represented in RDF referencing domain ontologies and queried through their SPARQL Endpoints. The primary cost is the creation of an ontology for each data source, as query resolution requires an entire merging ontology. Only people who understand domain-specific vocabulary can develop ontologies. Finding qualified ontology developers can be a costly endeavour.
- Costs of implementation may be connected with the dedicated computers needed for data storage and retrieval. As depicted in 17, these computers contained between 8GB and 16GB of RAM. 60 GB was the smallest solid-state disc (SSD) that could store the selected data source. Notably, the table 3 RDF's storage analysis determined the quantity of data that could be stored on a single storage disc. For example, a 60GB SSD might save six months of beach RDF data. The data provider can then service the gadget every six months. As previously noted, the total cost was established prior to implementation by computing the overhead. Although the employment of low-cost computing devices contributes to the effectiveness of this system, the cost may be raised by human resources' labour time.

4.7 Use Cases

Our proposed on-demand data model is applicable in various industries. It offers practical and cost-effective solutions to SMEs. Businesses can build various innovative services enriched with machine learning and AI that respond to end-users personally. In comparison, legacy data trading constrained businesses to acquire bulky datasets that may incur more charges and require high maintenance. (i.e., filtering to process relevant data records). This study's notional use cases concern the tourism and housing industries. The former is a small company that enables consumers to browse and

	No Rule (ms)	SWRL (ms)
count	100.00	100.00
mean	107.40	519.13
std	11.44	27.17
min	90.00	481.00
25%	100.00	498.75
50%	104.00	512.00
75%	113.00	530.75
max	162.00	615.00

(a)

	No Rule (ms)	SWRL (ms)
count	100.00	100.00
mean	34.45	45.00
std	3.78	6.28
min	29.00	35.00
25%	32.00	40.00
50%	34.00	44.00
75%	36.00	50.00
max	49.00	60.00

(b)

	No Rule (ms)	SWRL (ms)
count	100.00	100.00
mean	130.04	105.68
std	33.95	37.55
min	80.00	55.00
25%	104.00	77.50
50%	125.00	96.00
75%	147.75	131.75
max	242.00	291.00

(c)

Figure 23: Descriptive analysis for the three evaluations query response time (ms)- with and without reasoning rules inserted. Tables explore the datasets nature and summarise their contents.

Evaluations	Shapiro-Wilk H0 : Data follow a normal distribution. H1 : Data do not follow a normal distribution.				Kruskal-Wallis H0 :Two samples are related H1 :Two samples are not related		Mann Whitney H0 : Sample distributions are equal H1 :Sample distributions are not equal	
	P-value = 0.05							
	No Rule	Result	SWRL	Result	No Rule /SWRL	Result	No Rule /SWRL	Result
One	4.70508E-08	*Reject H0*	5.02E-06	*Reject H0*	0.000	*Reject H0*	0.000	*Reject H0*
Two	3.52513E-08	*Reject H0*	0.0006118	*Reject H0*	0.000	*Reject H0*	0.000	*Reject H0*
Three	4.14385E-05	*Reject H0*	4.94E-07	*Reject H0*	0.000	*Reject H0*	0.000	*Reject H0*

Figure 24: Evaluation Statistical Analysis

Evaluations	Average Query Time (ms)				
	SPARQL Query		Difference	%	
	No Rule	SWRL			
One	107.4	519.13	411.73	383	↑
Two	34.45	45	10.55	31	↑
Three	130.04	105.68	-24.36	-19	↓

Figure 25: Evaluations query average time comparison

book trips to local attractions, promoting sustainable travel. The latter is a state agency that recommends properties with considerably clean air features (i.e. properties located in less polluted and quiet areas).

4.7.1 Use Case 1: TripRecommerder

TripRecomender is a small business that enables end-users to plan green trips to local attractions. It predominantly aims for high-rated customer satisfaction by leveraging AI self-learning competencies. The company adopted the online chat application, named bots or Chatbots. An intelligent program relies on actual data to carry out a specific task. Here, TripRecommender is planning to train its chatbot to search and suggest local tourist destinations and sustainable travel solutions, increasing customer satisfaction and boosting the company's revenue. Well-trained bots can reduce the time and effort spent in monotonous trip planning. Triprecommnder considers users' requests and suggests tailored options based on previously known information. The company's chatbot requires sufficient relevant data to analyse and turn into meaningful information to accomplish this task. Relevant data records may exist in separate datasets- making it challenging to aggregate. With our on-demand data model, Triprecommneder can query and retrieve granular data records model, fit to train the bot. (e.g., ten years of data for either local beaches occupation rate or available rental bikes on weekends at the local docking stations). Subsequently, it recommends customised offers to and infers new events. For example, the SPARQL query in **Code Snippet 10** expresses how our model retrieves certain weekends information about (i) a local beach's name, services and occupancy rate, (ii) a museum's opening hours and an (iii) available rental bike location.

Code Snippet 10: Use Case 1 - TripRecommender

```
PREFIX : <http://api.stardog.com/>
PREFIX stardog: <tag:stardog:api:>
PREFIX schema: <https://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ngsi-ld:<https://uri.etsi.org/ngsi-ld/default-context/>
PREFIX fiware:<https://uri.fiware.org/ns/data-models#>
PREFIX ngsi: <https://uri.etsi.org/ngsi-ld/>

SELECT *
    {?id a fiware:Beach;
    ngsi:name ?Name;
    fiware:facilities ?Facilities;
    fiware:occupationRate ?OccupationRate;
    fiware:Weekends    ?Weekends.

{SERVICE   <http://192.168.0.78:5820/Museum/query>
    {?id2 a fiware:Museum;
    fiware:openingHoursSpecification   ?OpeningHours.

{SERVICE   <http://192.168.0.128:5820/Bike/query>
    {?id3 a fiware:BikeHireDockingStation;
    fiware:availableBikeNumber ?availableBikeNumber;
    schema:address  ?address;
    ngsi:status ?Bikestatus. }}}}}
```

4.7.2 *Use Case 2 - CleanAir Housing*

CleanAir Housing is a medium business that sells and rent-out residential properties. Recently, the company noticed a staleness and price drop for properties in highly polluted areas based on its sales records. Conversely, homes in quiet and less polluted areas are most likely to sell in a year. Air pollution could negatively impact health. It happens when particular gases and liquids particles are released in the atmosphere, forming PM2.5 and PM10 particles and elevating Carbon monoxide (CO) and Nitrogen Dioxide (NO2) pollutants levels - above the clean air legal limit. Sources of these toxic gases include vehicle exhaust, factories and domestic combustion. Measuring Air Quality Index (AQI) can assess the air quality in areas of interest. As a result, CleanAir Housing decided to integrate AI-Powered services to (i) predict sales forecasts based on historical data and air pollution levels, (ii) recommend the optimal price to match the expected value and (iii) hunt for local fast-selling homes. Several environmental data records are needed to calculate air quality and noise levels. Traditional ways to acquire such data are deploying thousands of sensors or purchasing bulky environmental datasets. Both options demand time and funds. Hence, further data mining, processing and analysis are required to extract valuable insights. Integrating our on-demand data model enables the filtering and extracting of the required metrics from multiple datasets, forming search data that are fit to train the company's AI. For instance, the SPARQL query script in **Code Snippet 11** combined metrics from air quality and noise level datasets to filter out the addresses in areas with low noise levels and good AQI.

5 Related Work

5.1 Data Trading and Marketplaces

The IoT data generated by connected smart devices are highly dynamic, heterogeneous, and geographically distributed. The value of such data peaks when shared appropriately between data owners and consumers [20]. Over the past decade, buying and selling datasets online trended to monetize data sharing [21]. With the rising number of datasets and consumers, challenges and opportunities emerged. On the one hand, data marketplaces cut down the time and labour required to mine desired datasets, helping to build more efficient city services [22]. On the other hand, the lack of granular data, market awareness, and privacy constraints [23] demotivated data traders [24, 25]. In other words, consumers had to buy the entire dataset even if they only needed one or two records from it- incurring more cost and computational powers - in addition to data privacy, which remains challenging and ambiguous [26]. Liu et al. [27] stated that selling no-alternative entire datasets and slow-changing datasets are the main shortcomings of the data marketplace. Subsequently, they built a collaborative environment for sharing views and computing the cost of dynamic IoT data using an algorithm that maximizes fairness in data trading on marketplaces [28]. Furthermore, recent research by Patrizi et.al. [29] sought to optimize data gain and rewards between data providers and consumers. Here, authors engaged citizens with their corresponding businesses with contracts that maximize the business incentives and the citizen's desired data

Code Snippet 11: Use Case 2 - CleanAir Housing

```
PREFIX : <http://api.stardog.com/>
PREFIX stardog: <tag:stardog:api:>
PREFIX schema: <https://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ngsi-ld:<https://uri.etsi.org/ngsi-ld/default-context/>
PREFIX fiware:<https://uri.fiware.org/ns/data-models#>
PREFIX ngsi: <https://uri.etsi.org/ngsi-ld/>

SELECT*
{SERVICE  <http://192.168.0.128:5820/AirQuality/query>;
    {?id a  fiware:AirQualityObserved;
    schema:address  ?address;
    fiware:dateObserved ?date;
    fiware:Precipitation ?Precipitation;
    fiware:Reliability  ?Reliability;
    fiware:WindDirection?WindDirection;
    fiware:AirQualityIndex ?AirQualityIndex;
    ngsi-ld:Co  ?Co;
    ngsi-ld:CO_Level ?CO_Level;
    ngsi-ld:No?No;
    ngsi-ld:Nox ?Nox;
    ngsi-ld:No2 ?No2;
    ngsi-ld:So2 ?So2.
 {SERVICE  <http://192.168.0.78:5820/Noise/query>
    ?id2 a  fiware:NoiseLevelObserved;
    fiware:DateObservedFrom ?DateObservedFrom;
    fiware:DateObservedTo ?DateObservedTo;
    fiware:frequencies ?frequencies;
    fiware:DataProvider ?DataProvider;
    ngsi:location  ?location;
    ngsi-ld:lAeq_d ?lAeq_d;
    ngsi-ld:lAmax ?lAmax.
    FILTER(xsd:float(?lAmax) > 0.72)
    FILTER(REGEX(?address, "A"))
    FILTER(REGEX(?CO_Level,"Low"))
    FILTER(xsd:integer(?AirQualityIndex) < 100)
    FILTER(xsd:integer(?Nox) < 100)}}}
    Limit10
```

Liu et al. focused only on computing data pricing relying on algorithms, while Patrizi et al. modelled and simulated optimization methods to obtain optimal data trade offers. Despite the efforts made by these researchers to provide fair deals in the data trade, the solutions proposed do not provide the integration of disparate data sources and the flexibility to create cost-effective data retrieval. Our approach allows retrieving partial data directly from diverse datasets by creating customized data orders. As a result, data consumers can negotiate the cost and pay for what they order, reducing the volume of data purchased and their cost.

Ren et al. [30] aimed to minimize the total IoT data cost through Datum, an architecture for geographically distributed online data. Datum achieved a near-optimal solution for data buying and deployment decisions. Similarly, we suggest a mechanism to acquire data from disparate sources. However, in comparison, our approach allows granular search and reasoning over data. Jung et al. [31] thought about filling the pricing gap with PRIVATA. Primarily, a private environment enables users to place data orders with intended prices and negotiate ceilings. Equally, data providers can either agree or respond with counteroffers. PRIVATA tackles only the pricing challenge and has not considered other aspects, such as the diversity of information in a data order. Significantly, Staiano et al. [32] showed that location information has a more significant value than media, texts, and applications; nonetheless, we believe that reasonable price for all types of data produces a healthy market and fosters confidence. In the same context, Tang et al. [22] attempted a data pricing model that adapted the MiniCom algorithm to calculate query charges based on sellers' views. Tang et al. pricing model may result in variable rates if different sellers have contradicting views on the same data.

Miranda et al. [33] traded sensor data online by setting up the so-called Sensmart. Their platform offered the extra mile by allowing consumers to manage the sensing device generating the data. Nevertheless, many factors contribute to data trading viability, including proper pricing and mutual agreement between concerned parties. Despite the market supply and demand for various IoT datasets, the willingness of sellers and buyers to trade plays a significant role in maximizing revenue [34]. Yu et al. [35] studied the market supply and demand trend and applied it to mobile data trading. More specifically, Yu et al. [34] modelled the relationship between users and mobile network operators as a two-level Stackelberg. In the first level, the operator sets the service charges that maximize the return. Every user played a buyer or seller role in the second level to optimize price and quantity. The model outcome suggested that overly high mobile service charges may harm trader returns and user incentives. The silver lining in the analysis was

the users' sufficient awareness of data usage, where results suggested positive gain. Comparably, our method delegates the data pricing role to the data broker, standardizing the charges for both data sellers and buyers.

5.2 Smart City Data Modelling

Typically, sensors' datasets are aggregated and advertised on online platforms using different integration methods in the data marketplace. For example, De Virgilio et al. [36] introduced a semantic data management initiative named NYAYA, which can ingest a massive volume of semantically modelled datasets. Nevertheless, it provided scalable storage and competent querying answering and reasoning capabilities. The system developed by Fernandez et al. [25] targeted siloed data's discovery, integration and sharing. Our semantic data modeling differs from [36] and [25]. We transformed the datasets into RDF and queried them remotely through their SPARQL endpoints. Our proposed system has a faster average response time than those achieved by NYAYA [36]

Accessing semantically integrated information is non-trivial. The complexity lies in connecting to remotely distributed systems and obtaining competent answers to users' queries. For instance, Le-Phuoc et al. [37] attempted to solve accessibility challenges with Linked Stream Middleware. This abstract layer employed semantic web technologies and SPARQL queries to achieve system interoperability. Consequently, it hides the tedious processing details from developers and users, easing the integration of time-sensitive information with relevant linked data. Our proposed model, in contrast, offers access to semantically curated data in their source without the need to copy and move the data. Typically, semantically modelled data are evaluated through competency questions. These questions are formulated as SPARQL queries. Then the query outcome can be assessed for completeness, correctness and, more importantly, the response speed. Lopez et al. [38] evaluated the competency of answers obtained from using SPARQL queries and reasoning engine. The authors' research challenge was interpreting the users' questions - expressed in natural language - and transforming them into semantic queries that accurately and promptly fulfil the requests. Conversely, our research challenge faced creating complex SPARQL to answer the users' questions with respect to reasoning rules. We then evaluated the efficiency of our approach- focusing on the response time of a complex query that retrieves information from not only disparate sources but also reason over queried data.

Fisteus et al. [39] contributed to the literature a scalable and fast middleware named Ztreamy. Equipped with an API for data sharing and semantic filtering services, Ztreamy demonstrated that one server could publish real-time information to a maximum of 40K clients in a timely competing manner. In smart cities, data generated by IoT devices are often stored in designated databases depending on their type and format. The most common domains range from transport and environment [40] to tourism [41]. Each of these produces and manages its data autonomously. Research suggested that if a particular smart city's different domains (e.g., transportation, weather, noise pollution) linked their data together, more informed decisions could be made, and new events could be inferred. Hence, a justified need exists for online platforms that integrate heterogeneous data sources and present them in a user-friendly format. Another example is that Kotoulas et al. [42] implemented an online semantic platform, SPUD, to discover, model, and link dynamic and static data incrementally. SPUD was demonstrated in real-time traffic information processing from Dublin incorporating social media data. Lécué et al. [43] explored, analyzed and reasoned STAR-CITY, a semantic traffic system deployed in Dublin (Ireland), Bologna (Italy), Miami (USA) and Rio (Brazil). The authors explained the system structure and focused thoroughly on its constrained elasticity and ability to safely scale and replicate to other cities. Le-Phuoc et al. [44] addressed the IoT data integration challenge using linked data. The developed system called Graph of Things (GoT) ingested massive streams of heterogeneous data allowing access and query via SPARQL endpoint. Ali et al. [45] built a semantically enriched experience to detect events and perform reasoning and data analytics over the heterogeneous and distributed data streaming from IoT-enabled Communication Systems. While this research attempts semantically modelling smart city sensor data as a service, our research leveraged semantic modelling to trade such data with SMEs)

5.3 Semantic Web and Linked Data Applications

Semantic Web defines common standards for integrating data from heterogeneous sources employing tools including (i) Resource Description Framework (RDF) that offers the groundwork for linking data and publishing data on the web, (ii) Web Ontology Language (OWL) to build vocabularies, or ontologies, (iii) Simple Knowledge Organisation System (SKOS), a domain using RDF for implementing standards to aid using knowledge organization systems, for example, thesauri, taxonomies. Many researchers wrote about the use of the semantic web in this context. On the one hand, Hitzler [46] argued that semantic web technologies serve as a cost-effective approach for data discovery, publishing and reuse. On the other hand, Stephan et al. [47] stressed the efficiency of such technologies in achieving transparency. Instead, Shadbolt et al.[48] stated that heterogeneous data are more useful when semantically integrated using structured machine-understandable formats like Resource Description Framework (RDF). RDF uses Uniform Resource Identifiers (URIs) to identify data and link it with other related data on the web. Hence, most available open

datasets are in CSV format and require users to convert them to RDF before publishing them on the web. Mahmud et al. [49] have implemented a semantic model that converts CSV data into RDF with rich semantics and uploads it to the web using Linked Open Data (LOD) principles that carefully follow the rules and recommendations of the World Wide Web Consortium (W3C). Thus, Fabian et al.[50] opt to extend the RDF model, focusing on the challenges that published open data faces in terms of the limitations of Application Program Interfaces (APIs), the search and browsing, the quality of extracted information, and the security and licensing standards. Relatively, one of our research novelty is the ngsi data model's synthetic creation, programmatic transformation to RDF.

Applications of semantic web and linked data thrived in many domains, including education [51, 52], environment [53, 47] and transportation [54, 55]. To mention a few, Kebler et al. [56] developed an ontology design pattern for space-time prisms to achieve data integration, reuse and infer new events across linked data. Notably, ontologies, the explicit knowledge base for shared concepts, form the backbone of the semantic web. They exist in three types; top-level, which contains general terms applicable to various domains, (ii) domain ontology, which standardizes discipline concepts and (iii) ontology design pattern, which serves a specific purpose. Although Kebler et al. design pattern ontologies are considered developed, further updates and validation eventually apply. More specifically, updating design pattern ontologies with the most recent terms and cross-referencing them with other well-defined ontologies makes them more adaptable and reusable. Nevertheless, these processes demand time and collective effort. Exciting work by Gil et al. [57] aimed to minimize this time-lapse by introducing an online platform for paleoclimate experts, allowing them to annotate and update ontologies as required. Likewise, our system aims to save time and obtain customized data. Therefore, the decentralized data setup and reasoning mechanism contribute to the Gil et al. method.

Similarly, the health sector has its share in semantic web applications. To mention just a few, Dragoni et al. [58] launched PerkApp, a customized platform to monitor citizens' lifestyles in real-time and promote healthy living. PerkApp used semantic web technologies to model and store data-fostering reasoning by linking these data with domain knowledge. These applications rely heavily on end-user feedback to sustain satisfaction, improve services, and recommend new products. In this aspect, linked open data (LOD) proved its efficiency in various practices. For example, Fogli et al. [59] developed a context-aware recommender system favouring customer satisfaction. It leverages the power of the semantic web and linked data to assist customers in planning their travel itinerary while making the most of earned points of interest. The system outperformed the state-of-the-art, achieving a higher accuracy score and usefulness rank. In industry, semantic web modelled, unified and shared heterogeneous resources in cloud [60, 61]. Here, we stress that our business model can provide its consumers with tailored data to train their Artificial Intelligence (AI) applications.

6 Discussion, Limitation and Future Recommendation

SynchroniCity data marketplace sells sensor data in bulk. Consumers interested in specific observations from different sensors (e.g., air quality and noise level) must purchase each sensor's dataset separately. Such a practice may incur more charges and cause a high-latency network. Our study extended SynchroniCity data marketplaces with semantic web technologies to allow consumers to pay for the sensor's information they need - instead of buying the entire dataset. Consumers can acquire multiple observations from various data providers to fulfil their orders. For example, finding nearby parking spaces and museum opening hours on a particular date and time. Our end-product consists of a user-friendly interface [62] with an interactive map and a semantic data model.

More specifically, we (i) built a novel semantic model. It encompassed an Urban Data Exchange Ontology (UDEO) and FIWARE synthetic datasets for six different providers. (ii) conducted three different experiments as shown in 17 to determine the most practical modelling and storage solutions for the IoT data marketplaces. (iii) evaluated the experiments to demonstrate the effectiveness of the semantic modelling and SWRL - using different SPARQL queries, answering related competency questions.

The evaluation results support the hypothesis that reasoning over distributed data sources could be the ideal architecture for the IoT data marketplace. Evidently, in evaluation one 4.2, querying after inserting rules took a long time, and the time-lapse between queries with and without rules is relatively slow. In evaluation two 4.3, the query time response was reduced dramatically compared to evaluation one. Worth mentioning that to make SWRL rules work, we got to insert the rule independently in each database, unlike the evaluation one, where we dealt with a single database. Even when we inserted SWRL in each database, it did not activate with the SERVICE KEYWORD. The query line had to be executed within the rules inserted in the database. Query line targeting rules and reasoning responded when calling the database internally in Stardog studio work space. In evaluation three 4.4, the requested data can be obtained remotely via HTTP, using the host IP address, port and database name. User query breaks into triple patterns that interrogate data source SPARQL endpoints for results. Figure 25 compared the average evaluation time between querying databases with and without SWRL. Evaluation one showed the highest difference in query response time

between queries. Unexpectedly, the average response time on SWRL databases was lower than without rules. Therefore, we can draw valuable insights from our semantic model results as follows:

Semantic modelling and reasoning: Extracting explicit information from IoT SynchroniCity datasets was challenging since these data lacked formal definitions for widely shared standards. Our semantic model transformed them into queriable triplestores. The adapters (codes) mapped the data to RDF while referencing the UDEO. We inserted abstract rules into the databases to trigger reasoning such as *Sunnydays and weekends*. Sunny days rule sets available rental bikes level low, assuming higher demand on such days. While the weekends' rule deduced high occupancy rates on local attractions such as beaches and museums. Reasoning quickened query response time in experiment three by reducing the search space while filtering out information adhering to the rules. SPARQL queries managed to retrieve granular and semantically enriched and reasoned information from different datasets, stored locally and remotely. As a result, customized data requests can be achieved at low costs.
Interoperability: We suggest that experiment three's approach is interoperable. SPARQL allowed remote access through its endpoints, achieving seamless data sharing between different RDF databases stored in heterogeneous machines.
***Edge computing*:** In experiment three, we distributed the RDF datasets on separate computers operating independently. Executing data on these edge computers satisfied the horizontal scaling property, provided storage capacity, allowed computational flexibility (i.e., semantic modelling), and maintained low network latency (i.e., transmitting query results instead of the whole dataset).

Limitations: Despite that, our semantic model slashed data price, reduced network latency, and cut down information overload in the SynchroniCity data marketplace- yet this approach has some drawbacks. In particular, *the pricing structure, data platform security, data quality, and safe dissemination.* The pricing structure of our model allows consumers to pay for desired information instead of an entire dataset. Although it costs less, working out the total price of an order could be a complex task. We deal with different data providers with different data tariffs, broker fees, and taxes, if any. Each of these has independent calculations and may change over time. Therefore, offering fixed and competitive charges is an open challenge. We recommend adding a self-configuring pricing model that standardizes and price-marks data records across independent stores. For example, set one reasonable price for each data record- automatically updating to match the data market's supply and demand, then adjust the broker fees to be a fair percentage of the total bill.

Regarding security, accessing and queering data stored in remote machines via HTTP pose risks. Stardog offers security options such as authentication and password encryption; so far, its default security settings are considered minimal for network communications. Therefore, we recommend using the Secure Sockets Layer (SSL) encryption when deploying Stardog in production mode. Concerning data quality, *Synthetic data* used in this study are consistent with good quality, while real sensors data may have errors and missing values. Hence, data quality should be carefully addressed to replicate our real-life study. We recommend using accurate machine learning algorithms and artificial intelligence to detect and automatically correct errors. The information retrieved to fulfil consumers' requests creates new datasets. These datasets have diverse sources collected by sensors owned by different stakeholders. Thus, publishing them may raise data ownership and privacy concerns. A remedy could be building a tool that (i) traces the data lineage and accurately identifies the owner. (ii) applies a GDPR-compliant privacy policy agreed upon by all parties (data buyer, seller/owner and broker).

7 Conclusions

Data marketplaces are a new category of online marketplaces. Therefore, they are not well-researched within the academic community or well-implemented within the industry. SynchroniCity represents the first attempt to deliver a Single Digital City Market for Europe by piloting its foundations at scale in 11 reference zones - 8 European cities and 3 more worldwide cities - connecting 34 partners from 11 countries over 4 continents. The primary aim is to cater for the data needs of consumers. Data marketplaces also emphasize vital challenges around data acquisition. Data marketplaces incentivize owners to share the gathered data and recover part of the acquisition costs. A fundamental issue of syntactic data marketplaces such as SynchroniCity is that they do not selectively provide a mechanism to buy data. It means data consumers have to buy the entire datasets data owners offer. We demonstrated the utility of annotating IoT data with semantics through experiments, allowing highly selective data querying. As a result, we converted the IoT data marketplace into a queriable data store. Moreover, we reduced data prices by allowing consumers to request less data selectively, only the relevant data they needed to achieve the task at hand. We used semantic web technologies to address the critical challenges in IoT data marketplaces without significantly burdening the compute infrastructure.

References

[1] UDX, 2022.

[2] Authors Asunción Gómez-pérez, Enrico Motta, and Mari Carmen Suárez-figueroa. Introduction to the NeOn Methodology Introduction to the NeOn Methodology. *Cycle*, (c), 2005.

[3] Asunción Gómez-Pérez and Mari Carmen Suárez-Figueroa. Scenarios for building ontology networks within the NeOn Methodology. *K-CAP'09 - Proceedings of the 5th International Conference on Knowledge Capture*, pages 183–184, 2009.

[4] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The NeOn Methodology framework: Ascenario-based methodology for ontologydevelopment. *Applied Ontology*, 10(2):107–145, 2015.

[5] Helena Sofia Pinto, Steffen Staab, and Christoph Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *ECAI*, 2004.

[6] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. 1997.

[7] York Sure, Steffen Staab, and Rudi Studer. *On-To-Knowledge Methodology (OTKM)*, pages 117–132. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[8] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[9] María Poveda-Villalón, Alba Fernández-Izquierdo, Mariano Fernández-López, and Raúl García-Castro. Lot: An industrial oriented ontology engineering framework. *Engineering Applications of Artificial Intelligence*, 111:104755, 2022.

[10] Krzysztof Janowicz, Armin Haller, Simon J.D. Cox, Danh Le Phuoc, and Maxime Lefrancois. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56:1–10, 2019.

[11] Garvita Bajaj, Rachit Agarwal, Pushpendra Singh, Nikolaos Georgantas, and Valérie Issarny. 4w1h in iot semantics. *IEEE Access*, 6:65488–65506, 2018.

[12] Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi, and Kerry Taylor. Iot-lite: A lightweight semantic model for the internet of things. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 90–97, 2016.

[13] Laura Daniele, Frank den Hartog, and Jasper Roes. Created in close interaction with the industry: the smart appliances reference (saref) ontology. In *International Workshop Formal Ontologies Meet Industries*, pages 100–112. Springer, 2015.

[14] Rachit Agarwal, David Gomez Fernandez, Tarek Elsaleh, Amelie Gyrard, Jorge Lanza, Luis Sanchez, Nikolaos Georgantas, and Valerie Issarny. Unified iot ontology to enable interoperability and federation of testbeds. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 70–75. IEEE, 2016.

[15] Jerry R Hobbs and Feng Pan. Time ontology in owl. *W3C working draft*, 27:133, 2006.

[16] The Enterprise Knowledge Graph Platform | Stardog, 2022.

[17] Jie Jiang, Riccardo Pozza, Nigel Gilbert, and Klaus Moessner. Makesense: An iot testbed for social research of indoor activities. *ACM Trans. Internet Things*, 1(3), jun 2020.

[18] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

[19] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[20] Fan Liang, Wei Yu, Dou An, Qingyu Yang, Xinwen Fu, and Wei Zhao. A Survey on Big Data Market: Pricing, Trading and Protection. *IEEE Access*, 6:15132–15154, 2018.

[21] Fabian Schomm, Florian Stahl, and Gottfried Vossen. Marketplaces for data: An initial survey. *SIGMOD Record*, 42(1):15–26, 2013.

[22] Ruiming Tang, Huayu Wu, Xiuqiang He, and Stephane Bressan. Valuating Queries for Data Trading in Modern Cities. *Proceedings - 15th IEEE International Conference on Data Mining Workshop, ICDMW 2015*, pages 414–421, 2016.

[23] Youssef Khazbak, Junpeng Qiu, Tianxiang Tan, and Guohong Cao. Targetfinder: A privacy preserving system for locating targets through iot cameras. *ACM Trans. Internet Things*, 1(3), jun 2020.

[24] Igor Bilogrevic and Martin Ortlieb. "If you put all the pieces together..." - Attitudes towards data combination and sharing across services and companies. *Conference on Human Factors in Computing Systems - Proceedings*, pages 5215–5227, 2016.

[25] Raul Castro Fernandez, Pranav Subramaniam, and Michael J. Franklin. Data market platforms: Trading data assets to solve data problems. *Proceedings of the VLDB Endowment*, 13(11):1933–1947, 2020.

[26] Allison Woodruff, Vasyl Pihur, Sunny Consolvo, Laura Brandimarte, and Alessandro Acquisti. Would a privacy fundamentalist sell their DNA for $1000... if nothing bad happened as a result? The Westin categories, behavioral intentions, and consequences. *SOUPS '14: Proceedings of the Tenth Symposium On Usable Privacy and Security*, pages 1–18, 2014.

[27] Ziyang Liu and Hakan Hacigümüş. Online optimization and fair costing for dynamic data sharing in a cloud data market. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1359–1370, 2014.

[28] Paolo Missier, Shaimaa Bajoudah, Angelo Capossele, Andrea Gaglione, and Michele Nati. Mind my value. pages 1–8, 2017.

[29] Nathan Patrizi, Eirini Eleni Tsiropoulou, and Symeon Papavassiliou. Health data acquisition from wearable devices during a pandemic: A techno-economics approach. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[30] Xiaoqi Ren, Palma London, Juba Ziani, and Adam Wierman. Datum: Managing Data Purchasing and Data Placement in a Geo-Distributed Data Market. *IEEE/ACM Transactions on Networking*, 26(2):893–905, 2018.

[31] Kangsoo Jung, Junkyu Lee, Kunyoung Park, and Seog Park. Privata. *Schriftwechsel mit Metternich. Erster Teil: 1803–1819*, pages 309–326, 2019.

[32] Jacopo Staiano, Nuria Oliver, Bruno Lepri, Rodrigo De Oliveira, Michele Caraviello, and Nicu Sebe. Money walks: A human-centric study on the economics of personal mobile data. *UbiComp 2014 - Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 583–594, 2014.

[33] Ricardo Miranda, Miguel L. Pardal, and António Grilo. Sensmart: Sensor data market for the internet of things. *Proceedings of the ACM Symposium on Applied Computing*, pages 739–746, 2020.

[34] Junlin Yu, Man Hon Cheung, and Jianwei Huang. Economics of Mobile Data Trading Market. *IEEE Transactions on Mobile Computing*, 2020.

[35] Junlin Yu, Man Hon Cheung, Jianwei Huang, and H. Vincent Poor. Mobile Data Trading: Behavioral Economics Analysis and Algorithm Design. *IEEE Journal on Selected Areas in Communications*, 35(4):994–1005, 2017.

[36] Roberto De Virgilio, Giorgio Orsi, Letizia Tanca, and Riccardo Torlone. Semantic data markets: A flexible environment for knowledge management. *International Conference on Information and Knowledge Management, Proceedings*, pages 1559–1564, 2011.

[37] Danh Le-Phuoc, Hoan Quoc Nguyen-Mau, Josiane Xavier Parreira, and Manfred Hauswirth. A middleware framework for scalable management of linked streams. *Journal of Web Semantics*, 16:42–51, 2012.

[38] Vanessa Lopez, Spyros Kotoulas, Marco Luca Sbodio, Martin Stephenson, Aris Gkoulalas-Divanis, and Pól Mac Aonghusa. QuerioCity: A linked data platform for urban information management. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7650 LNCS(PART 2):148–163, 2012.

[39] Jesus Arias Fisteus, Norberto Fernández García, Luis Sánchez Fernández, and Damaris Fuentes-Lorenzo. Ztreamy: A middleware for publishing semantic streams on the Web. *Journal of Web Semantics*, 25:16–23, 2014.

[40] Mark S. Fox. The semantics of populations: A city indicator perspective. *Journal of Web Semantics*, 48:48–65, 2018.

[41] Raphaël Troncy, Giuseppe Rizzo, Anthony Jameson, Oscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai Dominik Kuhn, Catalin Barbu, Matteo Rossi, Irene Celino, Rachit Agarwal, Christian Scanu, Massimo Valla, and Timber Haaker. 3cixty: Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 46-47:2–13, 2017.

[42] Spyros Kotoulas, Vanessa Lopez, Raymond Lloyd, Marco Luca Sbodio, Freddy Lecue, Martin Stephenson, Elizabeth Daly, Veli Bicer, Aris Gkoulalas-Divanis, Giusy Di Lorenzo, Anika Schumann, and Pol Mac Aonghusa. SPUD - Semantic processing of urban data. *Journal of Web Semantics*, 24:11–17, 2014.

[43] Freddy Lécué, Robert Tucker, Simone Tallevi-Diotallevi, Rahul Nair, Yiannis Gkoufas, Giuseppe Liguori, Mauro Borioni, Alexandre Rademaker, and Luciano Barbosa. Semantic traffic diagnosis with STAR-CITY: Architecture and lessons learned from deployment in Dublin, Bologna, Miami and Rio. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8797:292–307, 2014.

[44] Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Hung Ngo Quoc, Tuan Tran Nhat, and Manfred Hauswirth. The Graph of Things: A step towards the Live Knowledge Graph of connected things. *Journal of Web Semantics*, 37-38(2016):25–35, 2016.

[45] Muhammad Intizar Ali, Naomi Ono, Mahedi Kaysar, Zia Ush Shamszaman, Thu Le Pham, Feng Gao, Keith Griffin, and Alessandra Mileo. Real-time data analytics and event detection for IoT-enabled communication systems. *Journal of Web Semantics*, 42:19–37, 2017.

[46] Pascal Hitzler. A Review of the Semantic Web Field. *Commun. ACM*, 64(2):76–83, jan 2021.

[47] E G Stephan, T O Elsethagen, L K Berg, M C Macduff, P R Paulson, W J Shaw, C Sivaraman, W P Smith, and A Wynne. Semantic catalog of things, services, and data to support a wind data management facility. *Information systems frontiers*, 18(4):679–691, 2016.

[48] N. Shadbolt, K. O'Hara, T. Berners-Lee, N. Gibbins, H. Glaser, W. Hall, and M. C. Schraefel. Open Government Data and the Linked Data Web: Lessons from data. gov. uk. *IEEE Intelligent Systems*, 27(3):16–24, 2012.

[49] S. M. Hasan Mahmud, Md. Altab Hossin, Md. Rezwan Hasan, Hosney Jahan, Sheak Rashed Haider Noori, and Md. Razu Ahmed. Publishing csv data as linked data on the web. In Pradeep Kumar Singh, Bijaya Ketan Panigrahi, Nagender Kumar Suryadevara, Sudhir Kumar Sharma, and Amit Prakash Singh, editors, *Proceedings of ICETIT 2019*, pages 805–817, Cham, 2020. Springer International Publishing.

[50] Fabian Kirstein, Kyriakos Stefanidis, Benjamin Dittwald, Simon Dutkowski, Sebastian Urbanek, and Manfred Hauswirth. *Piveau: A Large-Scale Open Data Management Platform Based on Semantic Web Technologies*, volume 12123 LNCS. Springer International Publishing, 2020.

[51] Pengfei Wu, Fengjuan Ma, and Shengquan Yu. Using a linked data-based knowledge navigation system to improve teaching effectiveness, 2021.

[52] Faiza Bashir and Nosheen Fatima Warraich. Systematic literature review of Semantic Web for distance learning, 2020.

[53] Justine Flore Tchouanguem Djuedja, Fonbeyin Henry Abanda, Bernard Kamsu-Foguem, Pieter Pauwels, Camille Magniont, and Mohamed Hedi Karray. An integrated Linked Building Data system: AEC industry case. *Advances in engineering software (1992)*, 152:102930–, 2021.

[54] Asier Moreno, Asier Perallos, Diego López-De-Ipiña, Enrique Onieva, Itziar Salaberria, and Antonio D Masegosa. A novel software architecture for the provision of context-aware semantic transport information. *Sensors (Basel, Switzerland)*, 15(6):12299–12322, 2015.

[55] B Margan and F Hakimpour. Integration of mobile gis and linked data technology for spatio-temporal transport data model. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(4):721–724, 2019.

[56] Carsten Keßler and Carson J.Q. Farmer. Querying and integrating spatial–temporal information on the Web of Data via time geography. *Journal of Web Semantics*, 35:25–34, 2015.

[57] Yolanda Gil, Daniel Garijo, Varun Ratnakar, Deborah Khider, Julien Emile-Geay, and Nicholas McKay. A controlled crowdsourcing approach for practical ontology extensions and metadata annotations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10588 LNCS:231–246, 2017.

[58] Mauro Dragoni, Marco Rospocher, Tania Bailoni, Rosa Maimone, and Claudio Eccher. *Semantic technologies for healthy lifestyle monitoring*, volume 11137 LNCS. Springer International Publishing, 2018.

[59] Alessandro Fogli and Giuseppe Sansonetti. Exploiting semantics for context-aware itinerary recommendation. *Personal and Ubiquitous Computing*, 23(2):215–231, 2019.

[60] Cheng Xie, Hongming Cai, Lida Xu, Lihong Jiang, and Fenglin Bu. Resource Service Toward Cloud Manufacturing. 13(6):3338–3349, 2017.

[61] Niklas Petersen, Lavdim Halilaj, Irlán Grangel-González, Steffen Lohmann, Christoph Lange, and Sören Auer. Realizing an RDF-based information model for, a manufacturing company – A case study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10588 LNCS:350–366, 2017.

[62] IOT Garage / Quarriable Smart City Data Market - User Interface · GitLab, 2022.

A Appendix

B Ontology Requirements Specification Document(ORSD)

B.1 Purpose

The Urban Data Exchange Ontology (UDEO) aims to describe sensor data in IoT marketplaces.

B.2 Scope

Internet of Things (IoT).

B.3 Implementation Language

The Web Ontology Language (OWL2).

B.4 Intended End-Users

- Small to Medium Enterprise (SME).
- Data Scientists.
- Computer Scientists.

B.5 Intended Uses

- To build a linked data that offers data on-demand (i.e., granular data retrieval from disparate sources).
- For reasoning about the data of interest.
- Build Artificial Intelligence (AI) models.

B.6 Ontology Requirements

B.6.1 Non-Functional Requirements

- UDEO must include IoT concepts such as sensor.
- UDEO must include relationships between IoT concepts and features of interest in space and time.

B.6.2 Functional Requirements

- Nine Competency Questions (CQs) formulated as SPARQL queries.

CQ1: *What is the temperature of the room at a given date and time?*

```
SELECT (AVG(?result) as ?averageHeartRate)
 {GRAPH <http://smarthome> {
        ?s rdf:type sosa:Observation;
        sosa:hasFeatureOfInterest ?fi;
        sosa:madeBySensor :BpmCore;
        sosa:observedProperty :heartRate;
        sosa:resultTime ?dt;
        sosa:hasResult [
             rdf:type qudt-1-1:QuantityValue ;
             qudt-1-1:numericValue ?result ;
             qudt-1-1:unit qudt-unit-1-1:HeartBeatsPerMinute ].
    ?fi bot:hasStorey :Level1.
    :Level1 bot:hasSpace :Kitchen, :LivingRoom.
    {
        SELECT ?building (COUNT(?storey) as ?storeyCount)
    {
            ?building a bot:Building ;
                      bot:hasStorey ?storey.}
                      GROUPBY ?building}
FILTER(?fi = ?building && ?storeyCount = 3)
BIND(SUBSTR(str(?dt), 0, 11) as ?date)}}
```

Listing 1: SPARQL Query for Scenario 10

CQ 2: *What are the air quality data in certain addresses at given date and time?*

```
INSERT {?id a   fiware:AirQualityObserved;
                fiware:AirQualityIndex ?AirQualityIndex;
                fiware:Precipitation ?Precipitation;
                fiware:Reliability  ?Reliability;
                fiware:WindDirection ?WindDirection;
                ngsi-ld:Co  ?Co;        ngsi-ld:No ?No;
                ngsi-ld:Nox ?Nox;       ngsi-ld:No2 ?No2;
                ngsi-ld:So2 ?So2.}
Where {
        SELECT *
          {?id a    fiware:AirQualityObserved;
                fiware:airQualityIndex ?airQualityIndex;
                fiware:precipitation ?precipitation;
                fiware:reliability  ?reliability;
                fiware:windDirection ?windDirection;
                fiware:airQualityLevel ?airQualityLevel;
                fiware:dateObserved  ?date;
                ngsi-ld:CO  ?CO;       ngsi-ld:CO_Level ?CO_Level;
                ngsi-ld:NO ?NO;        ngsi-ld:NOx ?NOx;
                ngsi-ld:NO2 ?NO2;      ngsi-ld:SO2 ?SO2.

BIND(STRAFTER(str(?airQualityIndex), ":") as ?Index)
BIND(STRBEFORE(str(?Index), ",") as ?AirQualityIndex)
BIND(STRAFTER(str(?precipitation), ":") as ?prec)
BIND(STRBEFORE(str(?prec), ",") as ?Precipitation)
BIND(STRAFTER(str(?reliability), ":") as ?rel)   BIND(STRBEFORE(str(?rel), ",") as ?Reliability)
BIND(STRAFTER(str(?windDirection), ":") as ?win) BIND(STRBEFORE(str(?win), ",") as ?WindDirection)
BIND(STRAFTER(str(?CO), ":") as ?carbon)         BIND(STRBEFORE(str(?carbon), ",") as ?Co)
BIND(STRAFTER(str(?NO), ":") as ?nitrogen)       BIND(STRBEFORE(str(?nitrogen), ",") as ?No)
BIND(STRAFTER(str(?NOx), ":") as ?nitOx)         BIND(STRBEFORE(str(?nitOx), ",") as ?Nox)
BIND(STRAFTER(str(?NO2), ":") as ?nitdi)         BIND(STRBEFORE(str(?nitdi), ",") as ?No2)
BIND(STRAFTER(str(?SO2), ":") as ?sulpher)       BIND(STRBEFORE(str(?sulpher), ",") as ?So2)}}
***********************************************
SELECT *
        {?id a  fiware:AirQualityObserved;
            schema:address  ?address;
            fiware:dateObserved ?date;
            fiware:Precipitation ?Precipitation;
            fiware:Reliability  ?Reliability;
            fiware:WindDirection ?WindDirection;
            fiware:AirQualityIndex ?AirQualityIndex;
            ngsi-ld:Co  ?Co;          ngsi-ld:CO_Level ?CO_Level;
            ngsi-ld:No ?No;           ngsi-ld:Nox ?Nox;
            ngsi-ld:No2 ?No2;         ngsi-ld:So2 ?So2.
     ?id2 a  fiware:NoiseLevelObserved;
            fiware:DateObservedFrom ?DateObservedFrom;
            fiware:DateObservedTo ?DateObservedTo;
            fiware:frequencies ?frequencies;
            fiware:DataProvider ?DataProvider;
            ngsi:location  ?location;
            ngsi-ld:lAeq_d ?lAeq_d;
            ngsi-ld:lAmax ?lAmax.

FILTER(xsd:float(?lAmax) > 0.72)   FILTER(REGEX(?address, "A"))
FILTER(REGEX(?CO_Level, "Low"))    FILTER(xsd:integer(?AirQualityIndex) < 100)
FILTER(xsd:integer(?Nox) < 100)}   Limit 10
```

Listing 2: SPARQL Query for Scenario 10

CQ 3: *Help me plan a day out!*

```
Select DISTINCT ?Type ?Facilities  ?Days
                ?ParkingName ?Category ?Temperature ?Distance ?PlacesNearby
{GRAPH <urn:ngsi-ld:Museum:121513>
{?id    ns1:museumType ?type.
 ?type          ngsi-ld:hasValue   ?Type.
 ?id            ns1:facilities     ?facilities.
 ?facilities    ngsi-ld:hasValue   ?Facilities.
 ?openingHours  ngsi-ld:hasValue   ?OpeningHours.
 ?OpeningHours  :dayOfWeek         ?Days;
                :opens             ?Opens;
                :closes            ?closes.}
GRAPH <urn:ngsi-ld:ParkingSpot:123456>
{ ?id2          ngsi-ld:name       ?name.
  ?name         ngsi-ld:hasValue   ?ParkingName.
  ?id3          ns1:category       ?catergory.
  ?catergory    ngsi-ld:hasValue   ?Category.}

sosa:Temperature sosa:hasValue     ?Temperature.
{ ?MuseumPoint   ns4:long          ?long;
                 ns4:lat           ?lat.
?PlacesNearby    geof:nearby (?MuseumPoint 10 unit:Kilometer);
                 ns4:long ?p_long;
                 ns4:lat ?p_lat;
BIND (geof:distance(sosa:MuseumPoint, ?PlacesNearby, unit:Kilometer) as ?Distance)}}
```

Listing 3: SPARQL Query for Scenario 10

CQ 4: *Where can I park and ride near a certain GPS location?*

```
SELECT *
    {?id a  fiware:ParkingSpot;
            fiware:category ?category;
            fiware:dataProvider ?dataProvider;
            ngsi:status ?status;
            ngsi:location ?location;
            ngsi:ParkingPoint ?ParkingPoint.
    ?id2 a  fiware:BikeHireDockingStation;
            fiware:availableBikeNumber ?availableBikeNumber;
            fiware:AvailableBikeNumber ?AvialableBikeNumber;
            fiware:SunnyDays ?SunnyDays;
            schema:address ?address;
            ngsi:status ?Bikestatus.
sosa:PointID a              pos:Point;
             pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance(?ParkingPoint, ?SOSAPoint, unit:Kilometer) as ?Distance)
FILTER(xsd:integer(?Distance < 500))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}
```

Listing 4: SPARQL Query for Scenario 10

CQ 5: *WeekDays Rule*

```
prefix rule: <tag:stardog:api:rule:>
[] a rule:SPARQLRule ;
    rule:content
```

```
    IF {
        ?id a fiware:AirQualityObserved;
            fiware:AirQualityIndex ?AirQualityIndex;
        BIND (?AirQualityIndex > 100  AS ?LowAirQuality)}
    THEN {
        Weekdays fiware:LowAirQuality ?LowAirQuality}.
```

Listing 5: SPARQL Query for Scenario 10

CQ 6: *we cycle everywhere! if the local beach is busy this weekend, we will head to the museum?*

```
SELECT *
        { ?id a fiware:Beach;
            ngsi:name ?Name;
            fiware:facilities ?Facilities;
            fiware:occupationRate ?OccupationRate;
            fiware:Weekends   ?Weekends.

{SERVICE <db://museum100k>
        {?id2 a fiware:Museum;
            fiware:openingHoursSpecification  ?OpeningHours.

{SERVICE <db://BikeHireDockingStation100k>
        {?id3 a fiware:BikeHireDockingStation;
             fiware:availableBikeNumber ?availableBikeNumber;
             schema:address             ?address;
             ngsi:status                ?Bikestatus.}}}}}
```

Listing 6: SPARQL Query for Scenario 10

CQ 7: *Where are the locations of available bikes?*

```
PREFIX ns1: <https://uri.fiware.org/ns/data-models#>
PREFIX ns2: <https://uri.etsi.org/ngsi-ld/>
PREFIX ns3: <https://schema.org/>

SELECT *
        {  ?BikeAvailable   ns1:freeSlotNumber ?availabenumber;
                            ns1:freeSlotNumber ?value.
           ?s               ns2:location       ?loc}
```

Listing 7: SPARQL Query for Scenario 10

CQ 8: *Where are the geographical information for available bikes near me?*

```
SELECT ?Result
        {   sosa:availableBikeNumber sosa:hasValue ?AvailabeBikeNumber.
            sosa:freeSlotNumber sosa:hasValue ?FreeSlotNumber.
            ?location a ngsi-ld:GeoProperty.
            ?address  :streetAddress   ?StreetAddress;
                      :addressLocality ?AddressLocality;
                      :adressCountry  ?Country .
           ?coordinate_node ns4:lat ?Lat;
                     ns4:long      ?Long.
BIND(CONCAT(STR(?StreetAddress),",", STR(?AddressLocality),",",STR(?Country)) AS ?Address).
BIND(CONCAT("POINT(",STR(?Long),"", STR(?Lat),")") AS ?Coordinates).
BIND(CONCAT("Available_Bike_No.","", STR(?AvailabeBikeNumber),"",
"Slot:", STR(?FreeSlotNumber),"Located_at","","Address:",
```

```
STR(?Address),",",  "coordinates␣:", STR(?Coordinates)) AS ?Result)}
```

Listing 8: SPARQL Query for Scenario 10

Scenario 9: *What is Barry Island beach profile and how can I get there?*

```
SELECT ?BeachInfo  ?Result

   {  sosa:availableBikeNumber  sosa:hasValue ?AvailabeBikeNumber.
      sosa:freeSlotNumber       sosa:hasValue ?FreeSlotNumber.
      ?location a               ngsi-ld:GeoProperty.
      ?address :streetAddress   ?StreetAddress;
               :addressLocality ?AddressLocality;
               :adressCountry   ?Country.
      ?coordinate_node ns4:lat  ?Lat;
                       ns4:long ?Long.
    sosa:accesssType (sosa:hasValue|ngsi-ld:hasValue)     ?accessType.
    sosa:beachType   (sosa:hasValue|ngsi-ld:hasValue)     ?beachType.
    sosa:facilities  (sosa:hasValue|ngsi-ld:hasValue)     ?facilities.
    sosa:occupationRate  (sosa:hasValue|ngsi-ld:hasValue) ?occupationRate .
    sosa:name        (sosa:hasValue|ngsi-ld:hasValue)     ?name.
    sosa:width        sosa:hasValue                       ?w.
    sosa:length       sosa:hasValue                       ?l.
    BIND(CONCAT(STR(?beachType),",", STR(?accessType)
    ,",", STR(?facilities),",", STR(?occupationRate)) AS ?BeachInfo).
    BIND(?w * ?l AS ?area).
    FILTER (?area >1000).
    FILTER REGEX (?name ,"(Barry)").
    FILTER REGEX (?BeachInfo, "(lifeGuard)").
    BIND(CONCAT(STR(?StreetAddress),",", STR(?AddressLocality)
    ,"," , STR(?Country)) AS ?Address).
    BIND(CONCAT(STR(?Long),":", STR(?Lat)) AS ?Coordinates).
    BIND(CONCAT("Available_Bike_No.","", STR(?AvailabeBikeNumber),"",
    "Slot:", STR(?FreeSlotNumber),"Locatedat","","Address:", STR(?Address)
    ,",",  "atacoordinates␣:", STR(?Coordinates)) AS ?Result )}
```

Listing 9: SPARQL Query for Scenario 10