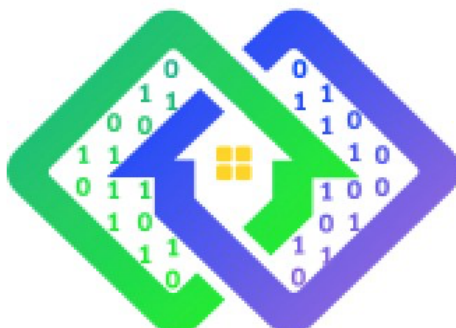


Grant Agreement N° 872592



PLATOON

Digital platform and analytic tools for energy

Deliverable D5.2 PLATOON Service Integration

Contractual delivery date:
M27

Actual delivery date:
31st of March 2022

Responsible partner:
P01: ENGIE, France

Project Title	PLATOON – Digital platform and analytic tools for energy
Deliverable number	D5.2
Deliverable title	Report on Service Integration
Author(s):	Hammad Aslam KHAN
Responsible Partner:	P01 – ENGIE
Date:	31.03.2022
Nature	R
Distribution level (CO, PU):	PU
Work package number	WP5 – Big data sharing and analysis reference implementations
Work package leader	UBO
Abstract:	The purpose of this document is to provide an end-to-end service integration for non-edge scenario of intelligence tool.

	This includes intelligence tool for non-edge scenario raw data extraction, data collection, semantic model's relevant concepts and its relationships, semantic transformation of data, and building of the end to end use case for reference implementation in WP5 package.
Keyword List:	Service Integration, Semantic pipeline, Data Analytics Toolbox, occupancy prediction, non-edge scenario, semantic data models, Templates

The research leading to these results has received funding from the European Community's Horizon 2020 Work Programme (H2020) under grant agreement no 872592.

This report reflects the views only of the authors and does not represent the opinion of the European Commission, and the European Commission is not responsible or liable for any use that may be made of the information contained therein.

Editor(s):	Hammad Aslam KHAN, Guillaume ARBOD
Contributor(s):	ENGIE, UBO, TIB
Reviewer(s):	Philippe CALVEZ (ENGIE) Maria-Esther Vidal (TIB) Erik Maqueda (TECN)
Approved by:	Philippe Calvez (ENGIE) – PLATOON Coordinator Erik Maqueda (TECN) – Technical Coordinator Eduardo Jimenez (IND) – Exploitation Coordinator
Recommended/mandatory readers:	WP2, WP6

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
V0.1	07/02/2022	First draft version	Hammad Aslam Khan
V0.2	14/03/2022	Added occupancy tool details	Carsten Draschner
V0.2	22/03/2022	Included the description of the Semantic Adapter implemented with the SDM-RDfizer	Maria-Esther Vidal
V0.3	22/03/2022	Added occupancy tool algorithmic details	Guillaume Arbod
V0.4	23/03/2022	Added details of occupancy tool as in WP5 demo	Farshad Bakhshandegan Moghaddam
V1.0	26/03/2022	Finalized version	Hammad Aslam Khan
V1.1	27/03/2022	Internal Review by TIB	Maria-Esther Vidal, Ahmad Sakor, Enrique Iglesias (TIB)
V1.2	27/03/2022	Addressing Reviews	Farshad Bakhshandegan Moghaddam
V1.3	29/03/2022	Addressing Reviews	Hammad Aslam Khan
V1.4	31/03/2022	Addressing final reviews	Hammad Aslam Khan Farshad Bakhshandegan Moghaddam Guillaume Arbod
V1.5	31/03/2022	Version submitted for approval	

Acknowledgements

The following people are hereby duly acknowledged for their considerable contributions, which have served as a basis for this deliverable:

Name	Partner
	ENGIE
	UBO
	TECN
	TIB

Table of Contents

TABLE OF CONTENTS.....6

LIST OF FIGURES.....7

LIST OF TABLES.....8

TERMS AND ABBREVIATIONS9

EXECUTIVE SUMMARY10

1 INTRODUCTION11

1.1 METHODOLOGY11

1.1.1 SAMPLE USE CASE IDENTIFICATION.....11

1.1.2 DATA PIPELINES CONSTRUCTION.....12

1.1.3 ANALYTICAL TOOL DEVELOPMENT13

2 DESCRIPTION OF USE CASE.....14

2.1 INTRODUCTION.....14

2.2 ANONYMIZED DATA INPUT15

2.3 SEMANTIC DATA MODEL.....15

2.3.1 SEMANTIC DATA MODELS OF BUILDING AND HVAC.....16

2.4 TOOL DESCRIPTION.....20

3 IMPLEMENTATION21

3.1 DEMONSTRATION SCENARIO.....21

3.1.1 DATA PIPELINE21

3.1.1.1 DATA STORAGE23

3.1.1.2 SCALABILITY24

3.1.2 ANALYTICAL TOOL – OCCUPANCY PREDICTION.....24

3.1.2.1 TOOL RESULT25

3.2 PRODUCTION SCENARIO27

3.2.1 DATA PIPELINE27

3.2.1.1 DATA INGESTION27

3.2.1.2 SEMANTIC TRANSFORMATION29

3.2.1.3 DATA STORAGE33

3.2.1.4 DATA QUERY33

3.3 ANALYTICAL TOOL – OCCUPANCY PREDICTION.....33

3.3.1 MACHINE LEARNING MODEL TRAINING.....34

3.3.2 THE FORECASTING PART34

3.3.3 TOOL RESULT.....34

4 CONCLUSION.....36

5 REFERENCES.....37

List of Figures

FIGURE 1 PIPELINE CONNECTING A SEMANTIC ADAPTER AND A FEDERATED QUERY PROCESSING ENGINE..... 13

FIGURE 2 OCCUPANCY PREDICTION TOOL IN CONTEXT OF PILOT3A 14

FIGURE 3 COMPARISON OF PRIVATE(RED) VS NON-PRIVATE(BLUE) DATA..... 15

FIGURE 4 A PORTION OF THE PLATOON SEMANTIC DATA MODEL FOR THE BUILDING PROPERTIES..... 17

FIGURE 5 HVAC RELATIONSHIP TO THE BUILDING IN DATA MODEL 18

FIGURE 6 EXTRACT OF HVAC SYSTEM..... 19

FIGURE 7 DATA INPUT AND OUTPUT FOR OCCUPANCY PREDICTION TOOL 20

FIGURE 8 PIPELINE DOCKERIZED COMPONENTS 21

FIGURE 9 THE PLATOON PIPELINE 22

FIGURE 10 COMMANDS TO CREATE THE DOCKER IMAGE OF THE PIPELINE..... 23

FIGURE 11 RESULT OF THE TOOL AS LINEAR PLOT 26

FIGURE 12 RESULT OF THE TOOL AS LINEAR PLOT (SHIFTED IN TIME)..... 26

FIGURE 13 RESULT OF THE TOOL AS DOT PLOT..... 27

FIGURE 14 DATA INGESTION PIPELINE SNAPSHOT OF ENGIE’S SEMANTIC ADAPTER IMPLEMENTATION 28

FIGURE 15 SPARQL GENERATE BASED SEMANTIC TRANSFORMATION CONFIGURATION 30

FIGURE 16 PREDICTED (ZONE LEVEL) VS ACTUAL OCCUPANCY (OVERALL SUM) DATA RESULT 34

FIGURE 17 RESULT OF OCCUPANCY PREDICTION WITH CONTINUOUS EXECUTION FOR A PERIOD OF ONE WEEK..... 35

List of Tables

TABLE 1 TERMS AND ABBREVIATIONS..... 9
TABLE 2 DATA SOURCES IN OCCUPANCY PREDICTION TOOL..... 19
TABLE 3: DATA STATISTICS 24
TABLE 4: RUNTIME IN SECONDS..... 24

Terms and abbreviations

AFNOR	French Standardization Association
ANSI	American National Standard Institute
BFO	Basic Formal Ontology
BOT	Building Ontology Topology
CA	Consortium Agreement
CIM	Common Information Model
CO	Confidential
DM	Dissemination Manager
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DSO	Distribution System Operators
EC	European Commission
EC/EP	Energy Consumption / Energy Production
ENTSO-E	European Network of Transmission System Operators
EV	Electric Vehicle
HVAC	Heating Ventilation Air Conditioning
IEC	International Electrotechnical Commission's
ISO	International Organization for Standardization
LLUC	Low Level Use Case
LLUC	Low Level Use Case
NTL	Non-Technical Losses
OEMA	Ontology for Energy Management Applications
OWL	Ontology Web Language
PV	Photovoltaic Panel
RDF	Resource Description Framework
RUL	Remaining Useful Life
SAREF	Smart Appliances REFerence ontology
SCADA	Supervisory Control And Data Acquisition
SDM	Semantic Data Model
SEAS	Semantic Energy Aware System
SSN	Semantic Sensor Network Ontology
SUMO	Suggested Upper Merged Ontology
TOVE	TOronto Virtual Enterprise
TSO	Transmission System Operators
UC	Use Case
WP	Work package
HVAC	Heating, Ventilation and Air Conditioning
ETL	Extract Transform Load
FQP	Federated Query Processing

Table 1 Terms and Abbreviations

Executive Summary

Task 5.2 develops methods for integration and deploying PLATOON services to test their behavior on the operational environment before they can be deployed for production use. Based on a sample use case, the deliverable focuses on two scenarios to explain service integration in end-to-end context:

1. Demonstration scenario for WP5 with more focus towards reference architecture validation
2. Production scenario in which implementation of reference architecture components is more focused towards production features like availability, scalability, complexity of data etc.

The deliverable explains the methodology adapted to a test use case of occupancy prediction to build the context for implementation identified from Pilot3a. Implementation is achieved through:

- i. Construction of data pipelines to prepare data required to test PLATOON services in respective environment (demo or production) that constitute data ingestion, data validation and data transformation to various formats;
- ii. Development of the occupancy prediction tool and results visualization on relevant dashboard for insights.

For the end-to-end scenario of the use case, where test deployment platform is discussed please refer to D5.5. Its data harmonization with respect to the data model is further explained in D5.3. Production deployment is exhibited in end-to-end perspective of Pilot3a in WP6/T6.1.

All components discussed in this deliverable are made open source and relevant references are included in respective sections to check the deliverables released source code on GitHub repositories.

1 Introduction

Task 5.2 develops methods for integration and deploying PLATOON services to test their behavior on the operational environment before they can be deployed for production use. The deliverable focuses on data pipelines to prepare data required to test PLATOON services in test environment and development of a service (analytical tool) that can run in a scalable way as well as how the data ingestion pipeline and the analytical tool is deployed in production real life scenario. End to end scenario discusses a sample use case from Pilot3a, i.e., Occupancy Prediction.

1.1 Methodology

In order to explain service integration in an end-to-end context, the deliverable focuses on explanation of a sample use case in two scenarios:

- 1) Demonstration scenario: in which reference architecture test platform is used that is delivered as an overall goal of WP5 and intends to test the analytical tool and perform desired performance benchmarks (scope of T5.5) for resource and availability requirements for production deployment.
- 2) Production scenario: that is target environment for implementation for the pilots in which implementation of reference architecture components is more focused towards production features like availability, scalability, complex data pipeline construction and clustering.

For each scenario:

- Sample use case is identified for which an analytical tool is developed to work on data and results are displayed on dashboard.
- Data pipeline is constructed for data ingestion, semantic transformation, query and storage.

1.1.1 Sample Use case Identification

To understand data processed and produced by the tool and how it is used, comprehending an end-to-end use case is pivotal. The HVAC (heating, ventilation, and air conditioning) optimization developed in pilot 3A framework provides an optimized operation schedule for each day of the week for the office building and its different zones based on the occupancy in the building and the comfort level required by the occupants. The HVAC optimization and control aims to:

- Optimize the building energy consumption;
- Maximize the comfort of occupants with the best energy efficiency; and
- Automate HVAC system control and reduce manual intervention on system controls.

The sample usecase selected for description is prediction of occupancy that serves as foundation for all other energy optimization tools. The HVAC systems follow a plan of operation that is configured by the building management system administrator. The goal of tools in pilot3a is to automate this scheduling of HVAC systems that is predicted based on forecast of building's occupancy. A building consists of zones from building operations point of view and heating, ventilation and cooling systems are switched ON/OFF with respect to zone as well.

Section 2 below presents is a detailed description of use case.

1.1.2 Data Pipelines Construction

As in any bigdata system, the PLATOON reference architecture contains desired components for ingesting data from a source system as part of ETL process and feeds this data to the tool. The PLATOON reference architecture comprises semantic adaptors for transforming data in different formats (e.g., CSV, JSON, or relational database) into an RDF knowledge base. Additionally, it includes a federated query processing (FQP) engine to evaluate SPARQL queries against the generated knowledge base. Figure 6 depicts a pipeline for the creation of knowledge graphs by transforming data different formats into RDF. This pipeline is agnostic of the knowledge base creation engine used to implement the semantic connector and the federated query processing engine. The semantic adapter converts data sources in different formats, e.g., CSV, JSON or relational databases, into RDF; this conversion is guided by mapping rules specified in a mapping language, e.g., SPARQL-Generate or RDF Mapping Language (RML). These rules define declaratively concepts (i.e., classes and properties) in the PLATOON semantic data models (presented in D2.3) in terms of data collected from the PLATOON data sources. As a result, during the execution of these rules, the semantic adapter populates classes and properties in a unified knowledge base. Then, the pipeline uploads this unified knowledge base into a federation of SPARQL endpoints (e.g., in Virtuoso or Fuseki). The federated query processing engine provides an interface to these endpoints; it enables users to explore the knowledge graph and retrieve the answers to a SPARQL query.

For the sample use case, separate technology stacks are used for demonstration and production scenario to construct the data pipeline stressing the strength of the architecture that is technology agnostic and reference implementation can be based on hardware resource and other processing requirements. In this data pipeline, semantic adapter implementation is discussed in two modes;

- For WP5 Demonstration scenario: SDM RDFizerⁱ is used that is based on RDF Mapping Language and purpose built for data semantic transformation
- For production scenario: Semantic adapter from Engie is used that is extension of opensource Apache NiFi solution that focuses on complete data pipeline (ETL) with a new processor created for semantic adaptation based on SPARQL generateⁱⁱ

Semantic adapter from ENGIE is made opensource as part of PLATOON deliverable and is published on PLATOON GitHub repositoryⁱⁱⁱ whereas RDFizer is already an opensource tool as well. Section 3.1.1.2 is a detailed description about data pipeline for demonstration scenario and section 3.2.1.2 is description of data pipeline for production scenario.

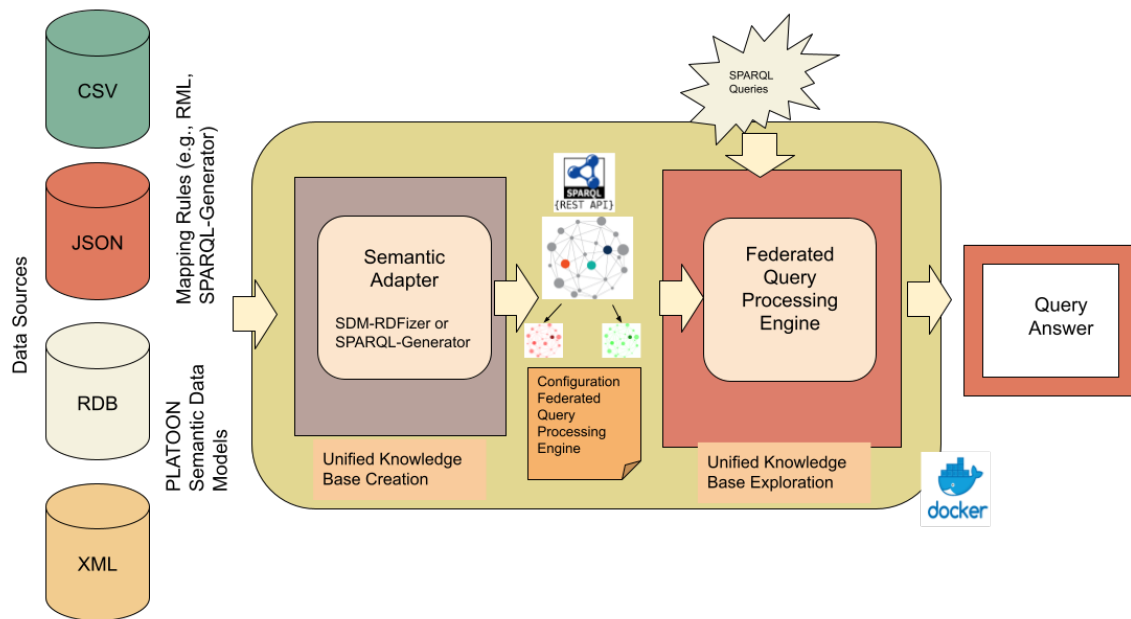


Figure 1 Pipeline connecting a Semantic Adapter and a Federated Query Processing Engine

1.1.3 Analytical Tool Development

An analytical tool within PLATOON is a machine learning or statistical tool that operates on input data and produces results. Produced result is used either as an input to the other tools or on analytical dashboard for insights. The selected sample usecase tool is implemented in two modes as well:

- For WP5 demonstration, basic tool is implemented whose further details are available in section 3.1.2.
- For production scenario, tool is implemented based on a parallelization framework. and is released as opensource implementation ready to execute on Apache Spark and Apache Flink. The tool uses SANSa transformers for using semantic data as direct input to the logistic regression model. Further details are available in section 3.2.2 of same document.

2 Description of Use case

Sections below explain end-to-end use case in which Occupancy prediction plays its role, description of raw data involved in pilot in general and in use case in particular

2.1 Introduction

Pilot 3A is related to the ENGIE Lab CRIGEN building office located in the Paris region. The office has a Building Management System (BMS) controlling the HVAC and comfort in different zones of the building

Two low-level use cases (LLUC) have been developed within the scope of this pilot:

- LLUC-3A-01-Optimizing HVAC control regarding occupancy.
- LLUC-3A-02-Provide demand response services through building inertia and HVAC controls.

Figure 2 shows the high-level flow diagram with input/outputs and the interaction of tools in the use case LLUC3A-01 that targets energy optimization by estimating the time for pre-heating/cooling.

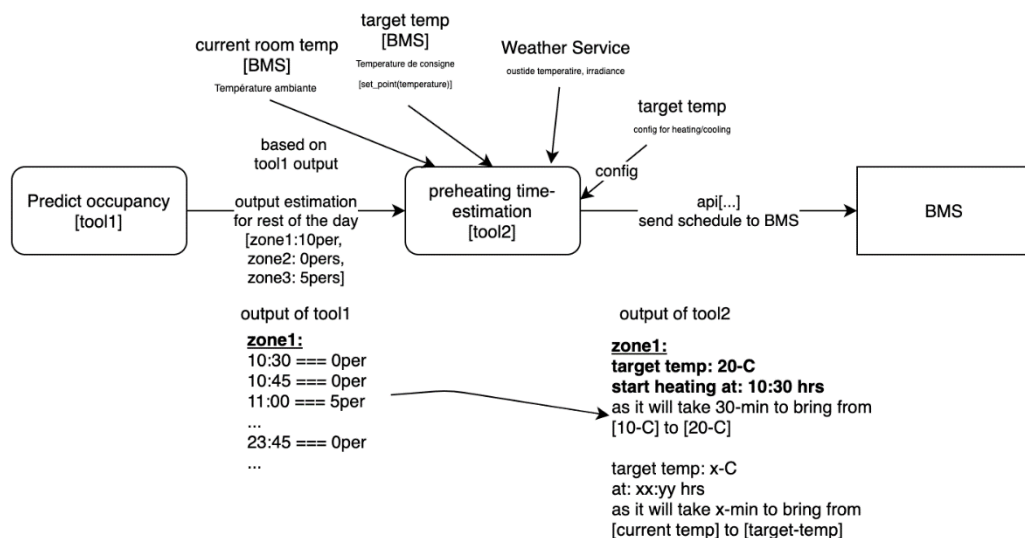


Figure 2 Occupancy Prediction Tool in context of Pilot3a

More details about the pilot and the specific use cases can be found in D1.1 and D6.1. The following sections explain in more detail the corresponding data analytics tools developed as part of Pilot3a use case involving occupancy data prediction

This complete use case is formed of the following data analytics tools defined in deliverable D4.1:

- Occupancy prediction in the building; and
- Preheating and precooling time estimation.

However, WP5.2 is about first part of above use case, i.e. Occupancy Prediction in the building.

2.2 Anonymized Data Input

There are three main sources of data in the Pilot3a occupancy prediction usecase context:

- WIFI Connections:
- LAN Connections; and
- Computer vision tool based on edge device installed on entrances and exits of the building (used only for demonstration purposes for IDS at the edge).

Data received from WIFI and LAN connections of the building is first cleaned by identifying nodes that are always present like printers, data centre servers etc, and then, it is used to count unique number of connections in building.

Due to proprietary nature of data, it has been anonymized using differential privacy^{iv} before sharing for PLATOON. Actual input data for the usecase is actual office building occupancy for a period of two months. Following are observations on differentially private data:

- Overall trend of 'private' data stays the same as that of 'non private' data.
- If on a certain day, some data elements, i.e. connections are not given as input for certain zone information, the result of Private data will not change.
- Privacy budget input if changed, will have direct impact on how much noise is induced (that has been carefully selected before data was exported in order to not change the trend of data pivotal for machine learning development).

Figure 3 is demonstration of a sample zone that highlights to what extent data has been distorted for anonymization purposes since the actual data cannot be shared for privacy reason. The graph shows the hourly sum of occupants for 24 hours in a single zone.

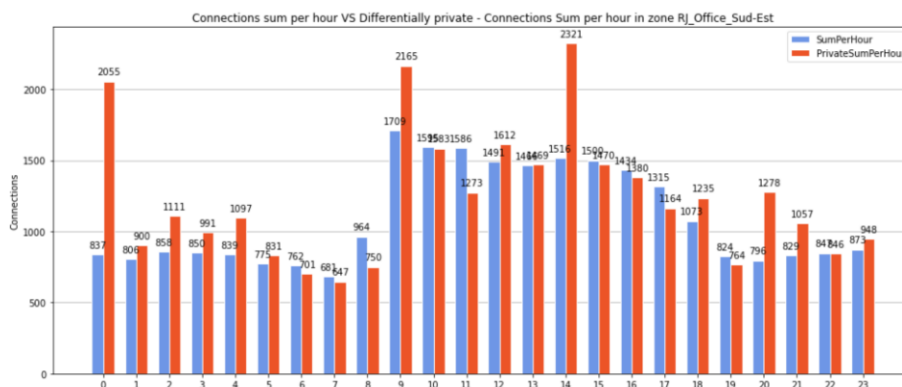


Figure 3 comparison of private(red) vs non-private(blue) data

2.3 Semantic Data Model

This section aims to describe the overall data model of Pilot3a and how building occupancy data concepts are related in overall building and HVAC data model.

2.3.1 Semantic data models of Building and HVAC

The pilot 3a aims to meet two goals in an office building. On the one hand, it aims to optimize the control of the HVAC (heating, ventilation and air conditioning) with respect to occupancy. On the other hand, it aims to provide a demand response service through HVAC control. To reach this purpose, two main elements are in this model: Building and HVAC. We present below an extract of our modelling of these two main elements.

Figure 2 represents an extract of Building and its occupancy and comfort properties from PLATOON data model of Pilot3a. The building concept is represented by *s4bldg:Building* from SAREF for building ontology which is equivalent to the concept *seas:Building* from the seas ontology. The building can be subdivided in different zones (*bot:Zone*) and is linked to these zones with the (*bot:containsZone*) relationship. Each zone (*bot:Zone*) has an occupancy property (*saref:Occupancy*). The zone and its occupancy property can be related with three types of occupancy relationships: the first one *plt:hasOccupancy* which is a generic property of occupancy that can be specialized on two sub properties *plt:hasClientsOccupancy* to indicate that the zone is occupied by customers and *plt:hasEmployeesOccupancy* to indicate that the zone is occupied by employees. The occupancy can be assessed either by some statistical algorithm or detected by occupancy sensor (*dogOnto:OccupancySensor*) which observes the occupancy property (*seas:ObservesProperty*). We can associate two types of evaluation with the occupancy. The first is the occupancy status (*plt:OccupancyStatusEvaluation*) to know if the zone is occupied or unoccupied. The second is the number of persons that occupy the zone (*plt:OccupiedNumber-Evaluation*). As each evaluation, these two evaluations have a temporal context of validity. For optimization of HVAC control, forecasting of the occupancy is important to schedule the cooling and the heating of the zones. This forecast is represented by *plt:ForecastOfOccupancy* concept that forecasts (*seas:forecastsProperty*) the *saref:Occupancy* Property. The optimization should respect for each *bot:Zone* the comfort level (*plt:ComfortLevelProperty*) which depends on (*plt:dependsOn*) *saref:Temperature* and *saref:Humidity*. This comfort level has four levels of evaluation (*seas:evaluation*) and those are *plt:ThresholdComfortLevelEvaluation*, *plt:MinimumThreshold ComfortLevelEvaluation*, *plt:BasicComfortLevelEvaluation* and *plt:BestComfortLevelEvaluation*

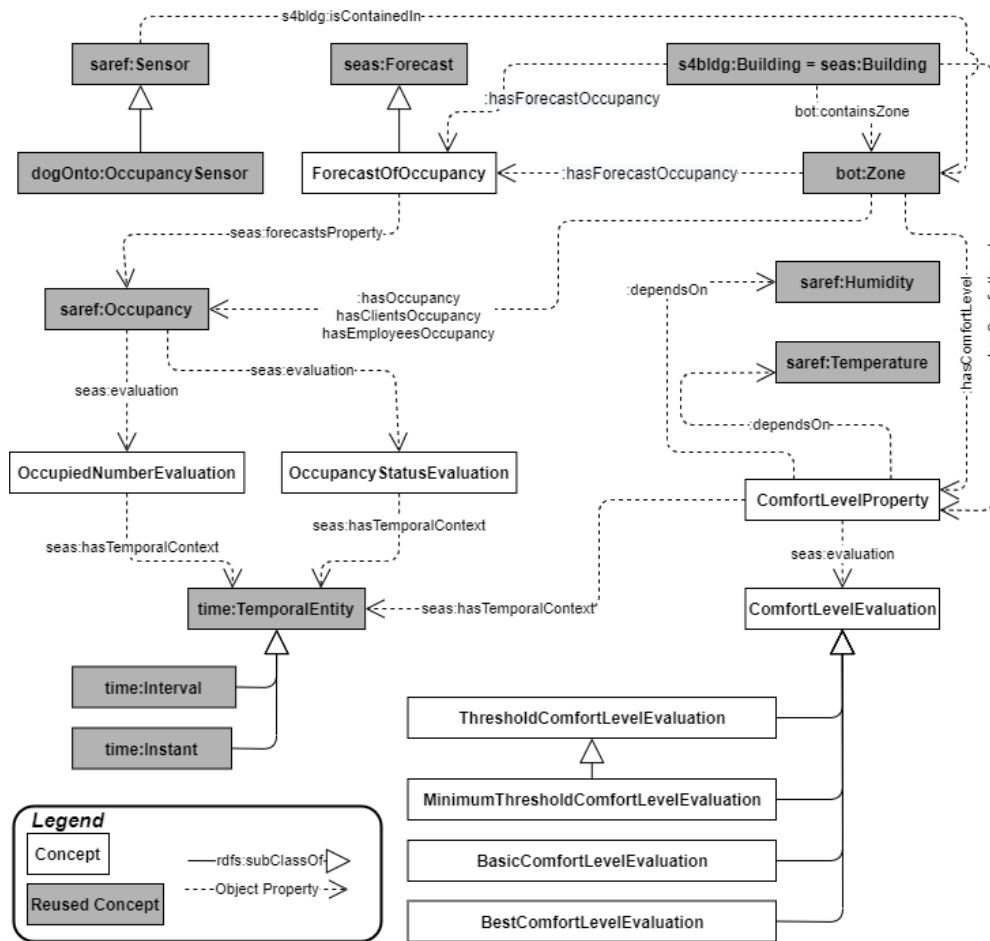


Figure 4 A portion of the PLATOON Semantic Data Model for the Building properties

For the HVAC domain, brick ontology^v was identified as a good candidate that covers a big part of the HVAC domain. However, after analyzing the taxonomy of this ontology, there was a modelling error detected concerning the use of the subsumption relationship (`rdfs:subClassOf`). Indeed, the subsumption relationship was confused with the part of (`part_of`) relationship. For example, the boiler, pump, fan, filter, and valve are all considered as HVAC whereas that are part of the HVAC system. Then, for brick concepts it is preferred to use the `rdfs:seeAlso` relationship. The HVAC is a system has three subsystems: the `plt:HeatingSystem` that is related by the relation `plt:hasHeatingSystem`, the `plt:CoolingSystem` that is related by the relation `plt:hasCoolingSystem` and the `plt:VentilationSystem` that is related by the relation `plt:hasVentilationSystem`. `Saref:HVAC` is located (`s4bldg:is-ContainedIn`) in `s4bldg:Building`. HVAC is controlled by a `plt:BuildingManagementSystem` that controls (`brick:controls`) the HVAC.

The cooling system and heating system has respectively `plt:CoolingValve` and `plt:HeatingValve` that are their subsystem (`seas:subSystemOf`). Each valve is located in (`s4bldg:is-ContainedIn`), a zone and has opening percentage property (`plt:OpeningPercentageProperty`) related by the relationship `plt:hasOpening Percentage`. A HVAC has three kind of operations (`plt:HVACOperationEvent`) that are a type of `pep:ProcedurExecution` and type of `plt:ScheduledEvent`. The operations are: (i) `plt:HeatingExecution` performed by (`ssn:madeByExecutor`) the `plt:HeatingSystem`. The heating execution can have two modes represented by `plt:hasMode` data property that can have the value “preheating” or “heating”. (ii) `plt:CoolingExecution` performed by (`ssn:madeByExecutor`) the `plt:CoolingSystem`. The cooling execution can have two modes represented by `plt:hasMode`

data property that can have the value “precooling” or “cooling”. (iii) `plt:VentilationExecution` performed by (`ssn:madeByExecutor`) the `plt:VentilationSystem`. Each of these kind of procedure executions has a temporal context (`seas:hasTemporalContext`) because that happened in a specific time.

Sometimes, in HVAC system there is also an `plt:AirHandlerUnit` that is a `seas:subSystemOf` the `saref:HVAC`. This air handler is connected to a heating unit, the cooling and the ventilation systems. A boiler (`plt:Boiler`) or heating coil (`plt:HeatingCoil`) can be `subSystemOf` of the heating system and the chiller (`plt:Chiller`) or cooling coil (`plt:CoolingCoil`) is a `subSystemOf` of cooling system. The `plt:Fan-coil` that has a `plt:Fan` and `s4bldg:Coil` as a subsystem, is connected to the air handler unit.

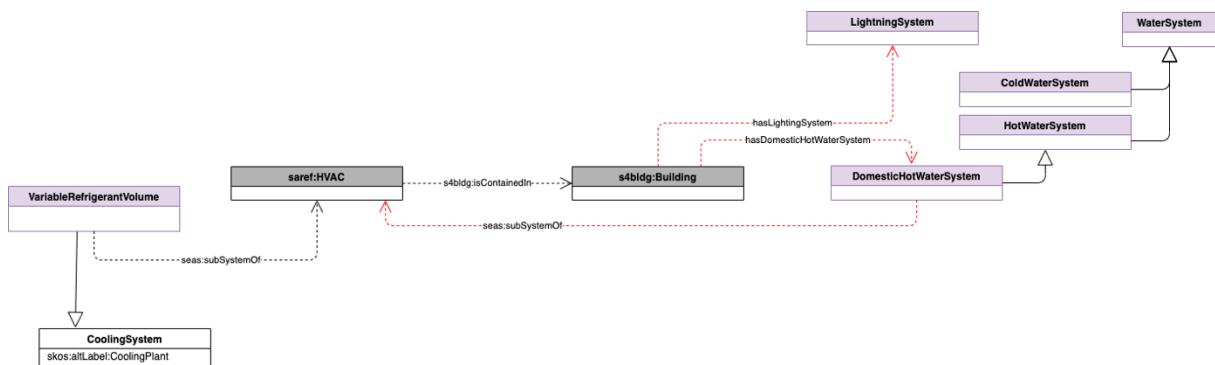


Figure 5 HVAC relationship to the building in Data Model

Some entities are added to the HVAC data model (figure 4) compared to the version of figure 3. This extension introduces a variable refrigerant volume (`plt:VariableRefrigerantVolume`) which is a Cooling system and a `seas:subSystemOf` the `saref:HVAC`. Also domestic hot water system (`plt:DomesticHotWaterSystem`), a subsystem of `saref:HVAC`.

2.4 Tool Description

Figure 7 shows the flow diagram for the Occupancy Prediction tool. As it can be seen, occupancy data is ingested from IT LAN and WIFI connections sources from each zone, cleaned up, aggregated, format converted and transformed to semantic data in pipeline.

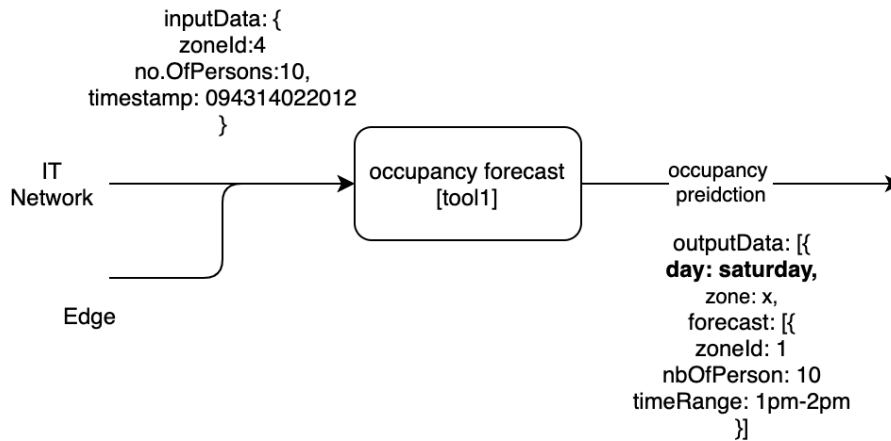


Figure 7 Data input and output for occupancy prediction tool

Whenever a new occupancy data is received, in real time mode, the tool updates the prediction for the next 48 hours from the time of execution. So, based on events, the produced results are transmitted (also as events) and stored in timeseries DB in part of Pilot3a but for the scope of task T5.2, data is exported as a file and in the pipeline its processed as a batch. Therefore, there are two execution modes of the tool in pilot3a.

Realtime:

In which the tool executes every 5 minutes on aggregated occupancy data received for building occupancy and produces next 48 hours prediction. Each time new data is received, already calculated occupancy prediction is overwritten such that at a given time, most recent prediction is available. This mode is currently in operation in Pilot3a and not in scope for WP5.

Batch:

In T5.2 scope, batch mode operates on occupancy data that is stored in semantic notation i.e., JSON-LD saved in DB. At the time of ingestion, the data is converted using semantic adapter and saved to DB using semantic notation. SANSAs transformers take as input the semantic data and convert to the format accepted by the model. Finally results of the batch execution mode are saved in DB directly. SANSAs^{vi} (Scalable Semantic Analytics Stack), uses Apache Spark^{vii} Apache Jena^{viii} and HDFS^{ix} to perform data analytics natively on large-scale RDF Knowledge Graphs (KGs). Through the development of SPARQLIFY^x and the developments within the PLATOON project, further modules were added to enable KG and regression-based batch processing for occupancy prediction. The modules are all open-source part of the SANSAs GitHub (<https://github.com/SANSAs-Stack>) repository and are scientifically documented by the following papers: Literal2Feature^{xi}, DistRDF2ML^{xii}, Semantic Web Analysis with Flavor of Micro-Services^{xiii}. Further information on the development and embedding of this analytics pipeline can be found in the deliverables D4.6, D5.1, and D5.5.

3 Implementation

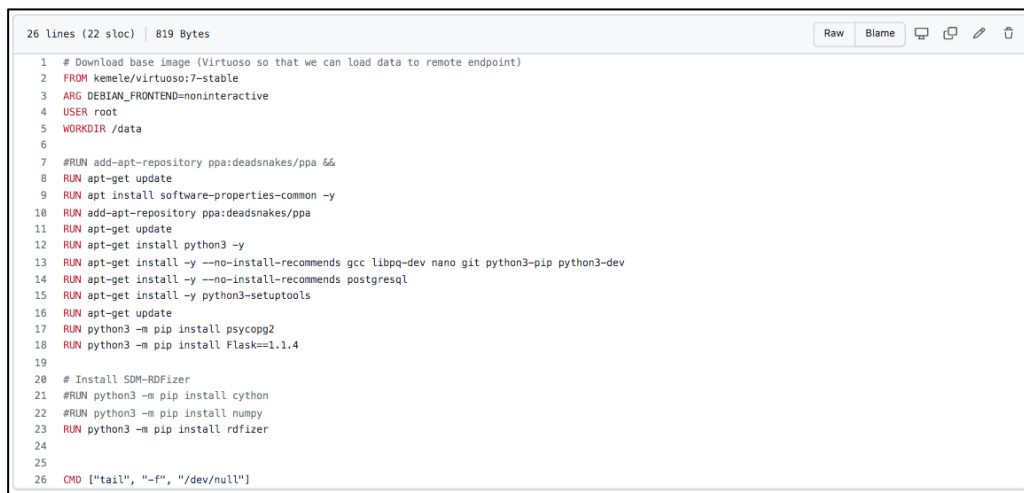
In order to explain service integration in an end-to-end context, the deliverable focuses on explanation of a sample use case in two scenarios:

1. **Demonstration scenario** that is based on test environment prepared for WP5 with more focus towards reference architecture validation
2. **Production scenario** that is based on production environment in which implementation of reference architecture components is more focused towards production features like availability, scalability, complexity of data, etc.

3.1 Demonstration Scenario

3.1.1 Data Pipeline

The pipeline is implemented as a bash script that executes a series of Docker images, each one implements a component of the pipeline. As a result, the pipeline is comprised of three Docker images: a) the SDM-RDFizer image; b) the Virtuoso images; and c) the FQP image. The pipeline is available in a GitHub repository¹ and uses dockerized components (Figure 7). The SDM-RDF^{xiv} is an open-source tool compliant with the W3C recommendation mapping language R2RML^{xv} and its extension the RDF Mapping Language (RML^{xvi}). SDM-RDFizer implement novel data structures, physical operators, and data compression techniques; these data management features, enable the efficient execution of pipelines of knowledge base creation; they also allow SDM-RDFizer to scale up to large and heterogeneous datasets.



```

26 lines (22 sloc) | 819 Bytes
1 # Download base image (Virtuoso so that we can load data to remote endpoint)
2 FROM kemele/virtuoso:7-stable
3 ARG DEBIAN_FRONTEND=noninteractive
4 USER root
5 WORKDIR /data
6
7 #RUN add-apt-repository ppa:deadsnakes/ppa &&
8 RUN apt-get update
9 RUN apt install software-properties-common -y
10 RUN add-apt-repository ppa:deadsnakes/ppa
11 RUN apt-get update
12 RUN apt-get install python3 -y
13 RUN apt-get install -y --no-install-recommends gcc libpq-dev nano git python3-pip python3-dev
14 RUN apt-get install -y --no-install-recommends postgresql
15 RUN apt-get install -y python3-setuputils
16 RUN apt-get update
17 RUN python3 -m pip install psycopp2
18 RUN python3 -m pip install Flask==1.1.4
19
20 # Install SDM-RDFizer
21 #RUN python3 -m pip install cython
22 #RUN python3 -m pip install numpy
23 RUN python3 -m pip install rdfizer
24
25
26 CMD ["tail", "-f", "/dev/null"]

```

Figure 8 Pipeline Dockerized Components

The GitHub repository contains required files and scripts for the execution of the PLATOON pipeline (Figure 9). These components are:

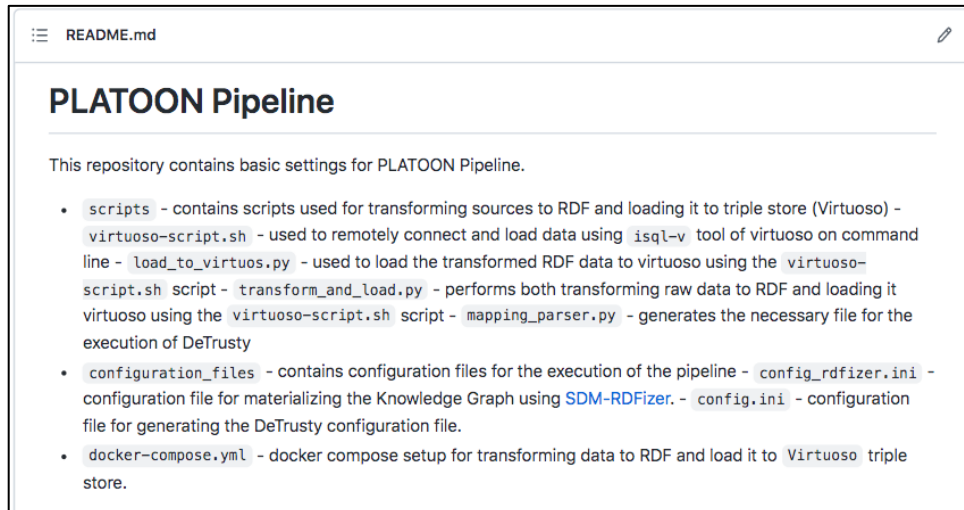


Figure 9 The PLATOON Pipeline

- **Scripts:** is a folder containing the scripts that transform mapping files and their corresponding data sources into RDF data, which is then loaded into the triple store (Virtuoso). These scripts are:
 - *Virtuoso-script.sh*: used to remotely connect and load data using the *isql-v* tool of Virtuoso.
 - *Load_to_virtuoso.py*: executes the *virtuoso-script.sh* to upload transformed RDF data to the Virtuoso triple store.
 - *Transform_and_load.py*: performs both the transformation of raw data into RDF data by invoking the SDM-RDFizer and uploads the data into a triple store by using *virtuoso-script.sh*.
 - *Mapping_parser.py*: generates the input file necessary for the execution of FQP by associating the classes from the mapping files with its corresponding predicates and endpoint that contains the RDF data.
- **Configuration_files:** is a folder that contains the configuration files that are necessary for the execution of the SDM-RDFizer and mapping_parser.py script.
- **Docker-compose.yml:** docker compose creates the docker images of the SDM-RDFizer, FQP, and Virtuoso. Figure 9 presents commands that composed the docker compose file.

```
51 lines (48 sloc) | 1.04 KB
1 version: "3.3"
2 services:
3   sdmrdfizer:
4     image: asakor/sdmrdfizer:4.0.1
5     hostname: sdmrdfizer
6     container_name: sdmrdfizer
7     domainname: platoon
8     volumes:
9       - ./data
10    networks:
11      - platoon
12    depends_on:
13      - pilot2akg
14    environment:
15      - SPARQL_ENDPOINT_IP=pilot2akg
16      - SPARQL_ENDPOINT_USER=dba
17      - SPARQL_ENDPOINT_PASSWD=dba
18      - SPARQL_ENDPOINT_PORT=1111
19      - SPARQL_ENDPOINT_GRAPH=http://platoon.eu/Pilot2A/KG
20      - RDF_DUMP_FOLDER_PATH=/data
21
22  detrusty:
23    image: prohe/detrusty:0.2.0
24    hostname: detrusty
25    container_name: detrusty
26    domainname: platoon
27    volumes:
28      - ./DeTrusty/Config:/DeTrusty/Config/
29    ports:
30      - "5000:5000"
31    networks:
32      - platoon
33    depends_on:
34      - pilot2akg
35
36  pilot2akg:
37    image: kemele/virtuoso:6-stable
38    hostname: pilot2akg
39    container_name: pilot2akg
40    domainname: platoon
41    volumes:
42      - ./rdf-dump:/data
43    ports:
44      - "8891:8890"
45      - "1116:1111"
46    networks:
47      - platoon
48
49  networks:
50    platoon:
51      external: false
```

Figure 10 Commands to create the docker image of the pipeline

FQP (a.k.a. DeTrusty) is a query engine that implements federated query processing against SPARQL endpoints. FQP is able to decompose a SPARQL query into subqueries composed of triple patterns that share the same subject. In case the query is posed against a federation of SPARQL endpoint, FQP selects a knowledge base(s) where each subquery will be executed. The partial results retrieved from the execution of each subquery are combined at the query engine level using non-blocking physical operators. Thus, FQP continuously generates query answers. Deliverable D2.8 provides a detailed description of the federated query engines integrated into the PLATOON framework; it also reports on the results of the performance empirical evaluation of FQP on Pilot 2a knowledge base.

3.1.1.1 Data Storage

RDF knowledge base created during the execution of the SDM-RDFizer is uploaded in the Virtuoso triple store (version 7.2.6) and is accessible via a SPARQL endpoint. Moreover, the configuration file required for the FQP execution is generated. Thus, RDF knowledge bases are available to be queried using both the Rest API service provided by the SPARQL endpoint and the one provided by the federated query engine.

3.1.1.2 Scalability

The above mentioned semantification pipeline has been used for the testing scenarios in D5.5. In that scenario, data related to Building occupancy prediction from Engie has been exploited. The data is in CSV format and contains the following columns:

- Id
- Date
- Hour
- Zone
- Source
- Connections

The “Source” column contains only the two values “WIFI” or “LAN”. The data has been anonymized and aggregated per hour. The file contains 31065 datapoints at a size of 1.4 MB. To show the performance of the mentioned semantification pipeline, some synthetic data from the original data has been generated to be able to put RDFizer and Detrusty under stress. The new data has been generated by shifting the date in the original data. The following table shows the statistics of the data.

Table 3: Data statistics

	Number of Rows	File Size	Knowledge Graph Size
Original Data	31066 ~ 30K	1.4 MB	30 MB
Synthetic Data 1	341716 ~ 350K	16.8 MB	300 MB
Synthetic Data 2	963016 ~ 1M	47.8 MB	1 GB

Data of various sizes can be used to measure the performance of the semantification process. The table below displays the runtime (in seconds) running RDFizer semantifer, loading data to triple storage, and querying FQP.

Table 4: Runtime in seconds

	RDFizer	Loading to TripleStore	FQP
Original Data	5.70s	4.10s	17.61s
Synthetic Data 1	40.74s	39.67s	251.62s
Synthetic Data 2	110.73s	116.58s	715.16s

As can be observed, increasing the size of the dataset increases the execution time of the semantification pipeline. It should be noted, however, that this development is sub-linear. That is, increasing the dataset size by a factor of ten does not increase the runtime by a factor of ten, but rather by a factor of less than ten.

3.1.2 Analytical Tool – Occupancy Prediction

The tool used in D5.5 (https://github.com/SANSA-Stack/SANSA-Stack/blob/feature/platoon2/sansa-ml/sansa-ml-spark/src/main/scala/net/sansa_stack/ml/spark/sparqlendpointreader/RegressionEngieUsecaseObject.scala) is a simplified version of the occupancy prediction done in pilot 3A. The goal of this simplified version is to provide a component that is ready for use in a demo environment

without the need for knowledge of the scenario in pilot 3A. Additionally, as defined in the PLATOON reference architecture, the components are required to provide a REST API. To adhere to this requirement, the demo occupancy prediction tool is built using these REST APIs as interfaces. This allows a user to later exchange the demo component with whatever tool they require. It is not the goal of the WP5 demo to evaluate the correctness of the outcome of the pilot 3A tool but rather validate the soundness of the PLATOON reference architecture.

In this tool, SANSa is used, which has the possibility to read data via a REST API directly from a Federated Query Engine. The input for this tool is:

- 1) HDFS Host IP
- 2) FDP Host IP
- 3) FDP Port
- 4) Sparql Query

These parameters are needed for SANSa to send the Sparql Query to the FQP and get back the result in HDFS and finally run the analytic. For the learning process, we used Random Forest as a regressor to predict the occupancy.

3.1.2.1 Tool Result

The analytic tool can consume data from any data source that provides the correct REST API. As such, the whole pipeline is flexible to swap out different components as needed. In our evaluation, we used the data sources as described in the previous section.

The figures below show the outcome of the tool by visualizing the predicted occupancy over time. The figures show the result of the tool in a Dashboard developed as open-source in task T5.4. The output confirms that the whole pipeline, including the demo tool, is working using REST APIs as communication between the components. **Error! Reference source not found.** shows the result of the tool as a linear plot. The green line at the bottom of the figure indicates the predicted occupancy over time. To study different time intervals, the tool allows a user to shift the plotted time frame as indicated in **Error! Reference source not found.**. The tool also provides further customization by changing for example from a linear plot to a dot plot, as shown in **Error! Reference source not found.**. Further explanation of the visualization part is presented in deliverable D5.4.



Select JSON url

Time Range

Select your dashboard

Time Lines

Add Dashboard

List of dashboard

Name	Params	
0	lines	prediction Delete

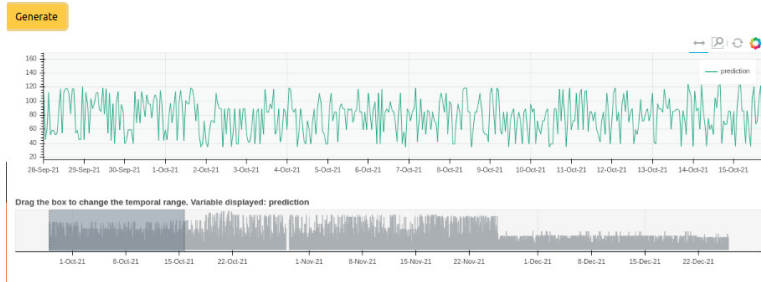


Figure 11 Result of the tool as Linear plot



Select JSON url

Time Range

Select your dashboard

Time Lines

Add Dashboard

List of dashboard

Name	Params	
0	lines	prediction Delete

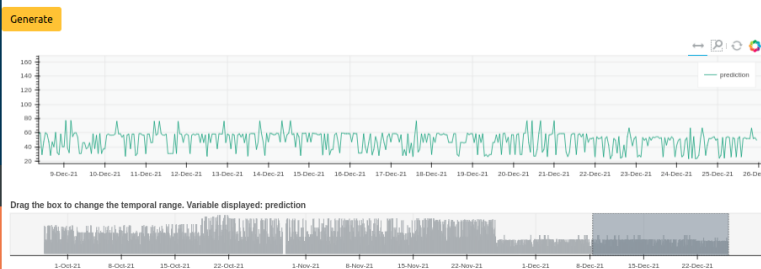


Figure 12 Result of the tool as Linear plot (shifted in time)

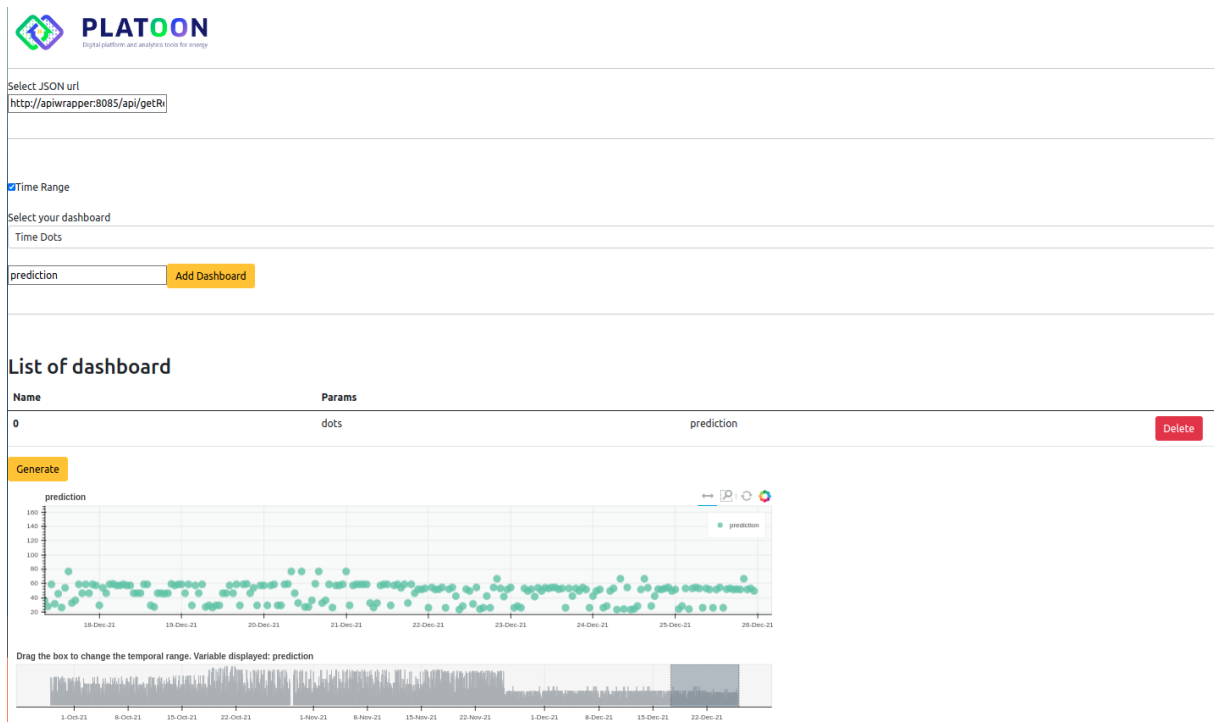


Figure 13 Result of the tool as dot plot

3.2 Production Scenario

3.2.1 Data Pipeline

The data pipeline with the whole ETL process is supported by semantic adapter ^{xvii} delivered by Engie that is capable of scaling to multiple instances supporting data provenance and fault tolerance as per D2.1 architecture description. At the same time, a data pipeline can be created that transfers same data at multiple streams automatically as soon as new data is available.

Engie's semantic adapter design is based on NiFi that is open source Apache project for data flow. NiFi's fundamental design concepts closely relate to the main ideas of Flow Based Programming [fbp] and hence is a natural choice of extension for development of semantic adapter targetted in the PLATOON reference architecture for all type of data treatment in a scalable way.

As a directed data flow graph, built-in processors can be used for various data cleaning/transformation purposes and in the pipeline, semantic adapter processor can be used as deemed necessary with respective sparql-generate query (sample data and flow is discussed in below sections with sparql-generate query).

3.2.1.1 Data Ingestion

Data ingestion process starts from reading the raw data file as soon as they are placed on certain location on network server (that could also be Amazon S3, SFTP server, etc) and its

storage to final destination after verification, format validation, cleaning, enrichment and semantic transformation. Below is the step by step description of data ingestion pipeline delivered as part of T5.2 scope in WP5:

1. Occupancy data generated by two sources (IT Wireless, IT LAN) are placed on Engie’s internal system.
2. Every five minutes new data is generated that is pulled into data pipeline but for WP5 demonstration purposes data is exported as a single file and placed on a location for ingestion.
3. Raw data is available as csv in the form:

```
.DARE,HOUR,ZONE,SOURCE,CONNECTIONS
0,2021-09-28,0,,LAN,3888
1,2021-09-28,0,,WIFI,9
```

4. As part of the data ingestion pipeline for the scope of WP5 batch data, its simply converted to JSON for demonstration of data cleanup and transformation (in pilot3a actual occupancy data pipeline data validation, data cleanup, enrichment etc. are additional steps that are performed while data is ingested) further described as part of D6.1 Pilot3a implementation

```
{
  "ZONE": "XYZ",
  "CONNECTIONS": 121,
  "TIMESTAMP": "202202151935",
  "SOURCE": "LAN"
},...
```

5. The input file contains all occupancy records over the period of time – those records are split into single records for transformation to semantic data based on JSON-LD data serialization (more detail in section 3.3.2).
6. Transformed data is now ready for storage on single or multiple data stores (current pipeline delivered supports storing to triplestore through HTTP for storage and data query through FQP and to HDFS for use by the occupancy batch tool for SANSA to load data directly).

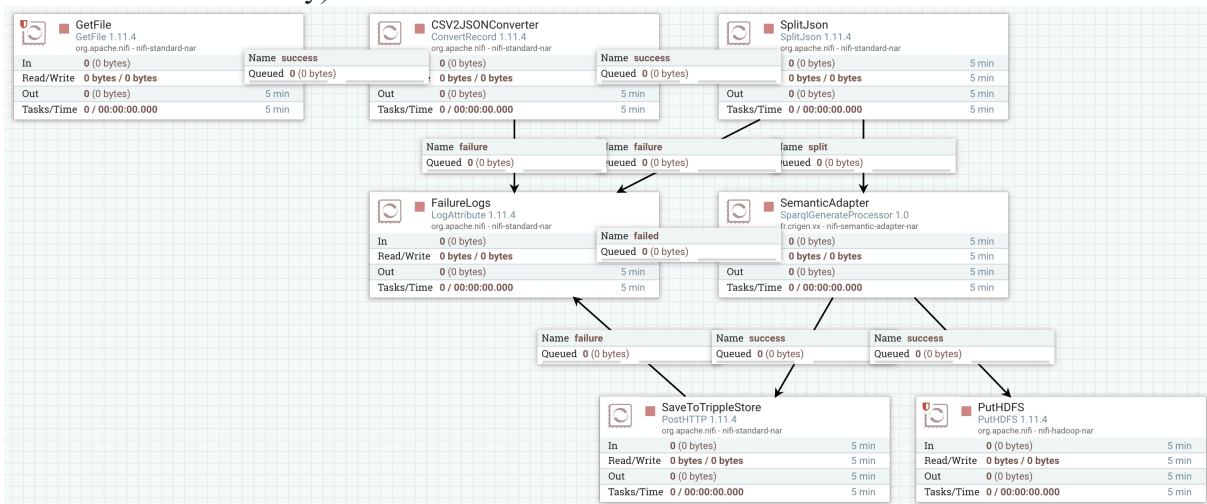


Figure 14 Data ingestion pipeline snapshot of Engie’s semantic adapter implementation

Figure 14 is a directed graph of data processors in which data passes as flowfiles and processed as single record. In semantic adapter, data flows are directed graph and its possible to use multiple semantic adapter processors in a single data flow depending on each aggregated or raw type data to be transformed. Scaling of semantic adaptation depends inherently on clustering capacity of Apache NiFi and hence is able to handle high velocity and volume of data as part of data flow. Even though semantic adapter based on NiFi is suitable for most of the scenarios for bulk data one time transformations, a separate semantic adapter batch is developed (that is not released as opensource at this stage).

Semantic transformation is done by one processor i.e. semantic adapter in the above directed graph data pipeline and explained further in next section. Each box in directed graph of figure 10 has statistics available in the UI that shows processing done every 5min:

In: input data flowfiles

Out: output data flowfiles

Read/Write: how many bytes of data are processed

Tasks/Time: How many tasks have been processed in how much time in time window of 5minutes

3.2.1.2 Semantic Transformation

Semantic adapter uses SPARQL Generate^{xviii} for generation of RDF data from JSON based occupancy data received. Semantic adapter supports multiple output formats for RDF data that can be configured in the processor as shown below in figure 11;

SPARQL-Generate^{xix} is an expressive template-based language to generate RDF streams or text streams from RDF datasets and document streams in arbitrary formats. It presents the following advantages:

- a. SPARQL-Generate leverages the expressivity of SPARQL 1.1: Aggregates, Solution Sequences and Modifiers, SPARQL functions and their extension mechanism;
- b. It can be extended to support new data sources and formats;
- c. It integrates seamlessly with existing standards for consuming Semantic Web data, such as SPARQL or Semantic Web programming frameworks.

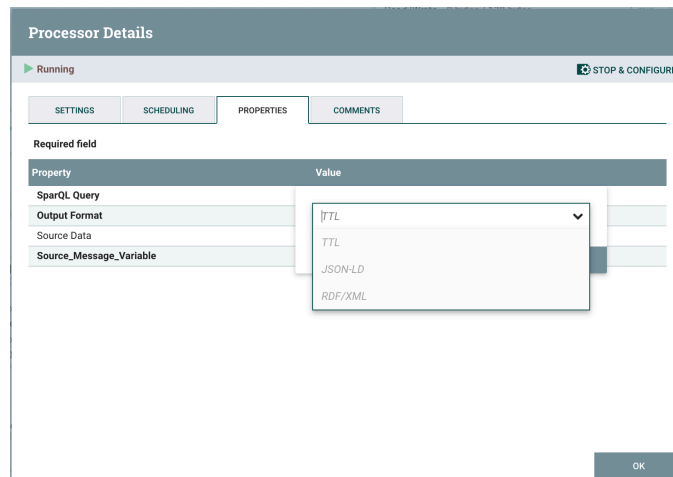


Figure 15 sparql generate based semantic transformation configuration

Using the SPARQL Generate mapping language, the execution of the following query transforms the input occupancy data characterized by zone name, number of occupants and timestamp and into instances of the PLATOON semantic data models. As a result, instance on an RDF graph are created.

```

BASE <http://engie.com/platoon/resource/building/>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX iter:<http://w3id.org/sparql-generate/iter/>
PREFIX seas:<https://w3id.org/seas/>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX plt:<https://w3id.org/platoon/>
PREFIX time:<http://www.w3.org/2006/time#>
PREFIX cdt:<http://w3id.org/lindt/custom_datatypes#>
PREFIX saref:<https://w3id.org/saref#>
PREFIX bot:<https://w3id.org/bot#>
PREFIX fun:<http://w3id.org/sparql-generate/fn/>
PREFIX prov:<http://www.w3.org/ns/prov#>
PREFIX pep:<https://w3id.org/pep/>

GENERATE {
?building a bot:Building;
    rdfs:label "CRIGEN-Stains";
    bot:containsZone ?zone.

?zone a bot:Zone;
    rdfs:label ?zoneID ;
    plt:hasPort ?port;
    plt:hasOccupancy ?occupancyProperty.

?occupancyProperty a saref:Occupancy, ?testPORT ;
    seas:isPropertyOf ?zone;
    seas:evaluation ?occupancyEvaluation.
?occupancyEvaluation prov:wasGeneratedBy ?sourceActivityURI, ?lanSourceActivityURI.
?sourceActivityURI a prov:Activity, pep:ProcedureExecution;
    rdfs:label ?sourceLabel.
?lanSourceActivityURI a prov:Activity, pep:ProcedureExecution;
    rdfs:label "LAN Connection".

GENERATE {
    ?occupancyEvaluation a plt:OccupiedNumberEvaluation;
        seas:hasTemporalContext ?temporalContext;
        seas:evaluatedSimpleValue ?{SUM(?occupiedNumberValue)}.
    ?temporalContext a time:Instant ; time:inXSDDateTime ?dateTime .
}
GROUP BY ?occupancyEvaluation ?temporalContext ?dateTime
WHERE {
# [{"CONNECTIONS":11,"TIMESTAMP":"20210622134500","ZONE":"R1_Office_Nord-Ouest","SOURCE":"WIFI"}]
BIND( fun:JSONPath( ?data, "$.ZONE" ) AS ?zoneID )
BIND( fun:JSONPath( ?data, "$.SOURCE" ) AS ?sourceID )
BIND( STR(fun:JSONPath( ?data, "$.TIMESTAMP")) AS ?date)
BIND( fun:JSONPath( ?data, "$.CONNECTIONS" ) AS ?occupiedNumber )
BIND( fun:JSONPath( ?data, "$.PORT" ) AS ?port )

```

```

BIND(fun:dateTime(concat(substr(?date, 1, 4), '-', substr(?date, 5, 2), '-', substr(?date, 7,2),"T",substr(?date, 9, 2),":",substr(?date, 11, 2),":00"),
"ISO_DATE_TIME")AS ?dateTime)
#BIND(IF(STRLEN(STR(?port))>0, "TRUE", "false") as ?testPORT)
BIND("http://engie.com/platoon/resource/building/" AS ?base)
BIND(URI(LCASE(CONCAT(?base,"CRIGEN-Stains"))) AS ?building)
BIND(IF(?sourceID="WIFI", URI(concat(LCASE(CONCAT(STR(?building),"/connection/wifi"))),
IF(?sourceID="LAN", URI(concat(LCASE(CONCAT(STR(?building),"/connection/lan"))), ?empty))AS ?sourceActivityURI)

BIND(IF(STRLEN(STR(?port))>0, URI(concat(LCASE(CONCAT(STR(?building),"/connection/lan"))), ?empty)AS ?lanSourceActivityURI)

BIND(IF(?sourceID="WIFI", "WIFI Connection",
IF(?sourceID="LAN", "LAN Connection", ?empty))AS ?sourceLabel)
BIND(IF(!BOUND(?zoneID), URI(LCASE(CONCAT(STR(?building),"/zone/unkown/port/",STR(?port))))),
URI(LCASE(CONCAT(STR(?building),"/zone/",STR(?zoneID)))) AS ?zone)
BIND(URI(LCASE(CONCAT(STR(?zone),"/property/occupancy"))) AS ?occupancyProperty)
BIND(URI(LCASE(CONCAT(STR(?zone),"/property/occupancy/evaluation",STR(?date)))) AS ?occupancyEvaluation)
BIND(URI(LCASE(CONCAT(STR(?zone),"/property/occupancy/temporalContext/",STR(?date)))) AS ?temporalContext)
BIND("{STR(?occupiedNumber)}"^^xsd:integer AS ?occupiedNumberValue )
}

```

Input data for semantic transformation

```

{
"CONNECTIONS":8,
"TIMESTAMP":20220323113000",
"ZONE":"R1_Office_Nord-Ouest",
"SOURCE":"WIFI"
}

```

Semantic transformed data (in json-ld format)

```

{
"@graph": [ {
"@id": "http://engie.com/platoon/resource/building/crigen-stains",
"@type": "bot:Building",
"label": "CRIGEN-Stains",
"containsZone": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete"
}, {
"@id": "http://engie.com/platoon/resource/building/crigen-stains/connection/wifi",
"@type": [ "pep:ProcedureExecution", "prov:Activity" ],
"label": "WIFI Connection"
}, {
"@id": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete",
"@type": "bot:Zone",
"label": "R2_Escalier_cafete",
"hasOccupancy": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete/property/occupancy"
}, {
"@id": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete/property/occupancy",
"@type": "safef:Occupancy",
"evaluation": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete/property/occupancy/evaluation/20220323113000",
"isPropertyOf": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete"
}, {
"@id": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete/property/occupancy/evaluation/20220323113000",
"@type": "plt:OccupiedNumberEvaluation",
"wasGeneratedBy": "http://engie.com/platoon/resource/building/crigen-stains/connection/wifi",
"seas:evaluatedSimpleValue": 2,
"hasTemporalContext": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete/property/occupancy/temporalcontext/20220323113000"
}, {
"@id": "http://engie.com/platoon/resource/building/crigen-stains/zone/r2_escalier_cafete/property/occupancy/temporalcontext/20220323113000",
"@type": "time:Instant",
"inXSDDateTime": "2022-03-23T11:30:00Z"
} ],
"@context": {
"evaluation": {
"@id": "https://w3id.org/seas/evaluation",
"@type": "@id"
},
"isPropertyOf": {
"@id": "https://w3id.org/seas/isPropertyOf",
"@type": "@id"
},
"containsZone": {
"@id": "https://w3id.org/bot#containsZone",
"@type": "@id"
},
"label": {
"@id": "http://www.w3.org/2000/01/rdf-schema#label"
},
"hasOccupancy": {

```

```

"@id" : "https://w3id.org/platoon/hasOccupancy",
"@type" : "@id"
},
"inXSDDateTime" : {
"@id" : "http://www.w3.org/2006/time#inXSDDateTime",
"@type" : "http://www.w3.org/2001/XMLSchema#dateTime"
},
"evaluatedSimpleValue" : {
"@id" : "https://w3id.org/seas/evaluatedSimpleValue",
"@type" : "http://www.w3.org/2001/XMLSchema#integer"
},
"hasTemporalContext" : {
"@id" : "https://w3id.org/seas/hasTemporalContext",
"@type" : "@id"
},
"wasGeneratedBy" : {
"@id" : "http://www.w3.org/ns/prov#wasGeneratedBy",
"@type" : "@id"
},
"owl" : "http://www.w3.org/2002/07/owl#",
"bot" : "https://w3id.org/bot#",
"saref" : "https://w3id.org/saref#",
"xsd" : "http://www.w3.org/2001/XMLSchema#",
"iter" : "http://w3id.org/sparql-generate/iter/",
"rdfs" : "http://www.w3.org/2000/01/rdf-schema#",
"seas" : "https://w3id.org/seas/",
"cdt" : "http://w3id.org/lindt/custom_datatypes#",
"plt" : "https://w3id.org/platoon/",
"pep" : "https://w3id.org/pep/",
"time" : "http://www.w3.org/2006/time#",
"prov" : "http://www.w3.org/ns/prov#",
"fun" : "http://w3id.org/sparql-generate/fn/"
}
}

```

RDF data (in Turtle format);

```

@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix bot: <https://w3id.org/bot#> .
@prefix saref: <https://w3id.org/saref#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix iter: <http://w3id.org/sparql-generate/iter/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix seas: <https://w3id.org/seas/> .
@prefix cdt: <http://w3id.org/lindt/custom_datatypes#> .
@prefix plt: <https://w3id.org/platoon/> .
@prefix pep: <https://w3id.org/pep/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix fun: <http://w3id.org/sparql-generate/fn/> .

<http://engie.com/platoon/resource/building/crigen-stains>
  a bot:Building ;
  rdfs:label "CRIGEN-Stains" ;
  bot:containsZone <http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest> .

<http://engie.com/platoon/resource/building/crigen-stains/connection/wifi>
  a pep:ProcedureExecution , prov:Activity ;
  rdfs:label "WIFI Connection" .

<http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest/property/occupancy/evaluation/20220323120001>
  a plt:OccupiedNumberEvaluation ;
  prov:wasGeneratedBy <http://engie.com/platoon/resource/building/crigen-stains/connection/wifi> ;
  seas:evaluatedSimpleValue 4 ;
  seas:hasTemporalContext <http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest/property/occupancy/temporalcontext/20220323120001> .

<http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest/property/occupancy>
  a saref:Occupancy ;
  seas:evaluation <http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest/property/occupancy/evaluation/20220323120001> ;
  seas:isPropertyOf <http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest> .

<http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest/property/occupancy/temporalcontext/20220323120001>
  a time:Instant ;
  time:inXSDDateTime "2022-03-23T12:00:00"^^xsd:dateTime .

<http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest>
  a bot:Zone ;
  rdfs:label "R1 Office_Nord-Ouest" ;
  plt:hasOccupancy <http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest/property/occupancy> .

```

The above SPARQL-Generate query has several parts;

- PREFIX : as in SPARQL

- GENERATE: statements are used to generate a triple in format <subject> <predicate> <object> with variable names

e.g. ?building a bot:Building;
 rdfs:label "CRIGEN-Stains";
 bot:containsZone ?zone.

- BIND: statements bind data received in JSON fields with the variable names assigned in GENERATE statements. In above GENERATE statement, ?building and ?zone are two variables that must be filled in from the data received

e.g. BIND("http://engie.com/platoon/resource/building/" AS ?base)
 BIND(URI(LCASE(CONCAT(?base,"CRIGEN-Stains"))) AS ?building)
 ...
 BIND(fun:JSONPath(?data, "\$.ZONE") AS ?zoneID)
 URI(LCASE(CONCAT(STR(?building),"zone/",STR(?zoneID)))) AS ?zone)

generates below RDF data (ttl) section by taking zoneId from \$.ZONE field of JSON data

```
<http://engie.com/platoon/resource/building/crigen-stains>
a          bot:Building ;
rdfs:label "CRIGEN-Stains" ;
bot:containsZone <http://engie.com/platoon/resource/building/crigen-stains/zone/r1_office_nord-ouest> .
```

Apart from above mentioned features, SPARQL-Generate query has many more interesting features (that are not used in this query) like ITERATORS that generate triples in batches, nested and modularized GENERATE clauses and most importantly FUNCTIONS on RDF terms, RDF graphs or SPARQL results written in LDScript^{xx}.

xxi

3.2.1.3 Data Storage

Delivered data pipeline is able to store transformed data on single or multiple data stores. Currently, the data pipeline released as opensource^{xxii} supports storing RDF data to triplestore through HTTP for storage, and data query through FQP and to HDFS for use by the occupancy batch tool for SANSA to load data directly.

3.2.1.4 Data Query

Query processing is done through the federated query processing and is scope of T2.4 and deliverables D2.4 and D2.8.

3.3 Analytical Tool – Occupancy Prediction

Occupancy prediction tool delivered by Engie is developed in coordination with UBO based on SANSA stack^{xxiii}. The tool is made scalable for high frequency data as its implementation is based on parallelization frameworks like Apache Flink (or Apache Spark) and is released as opensource^{xxiv}.

3.3.1 Machine Learning Model Training

The tool for occupancy prediction is based on deep learning model. The learning dataset is occupancy data for three weeks for the same building for which prediction needs to be carried out. A three-weeks period duration appears necessary to have a realistic characterisation of a particular occupancy status during the week. The learning dataset comprises the occupancy status and calendar variables (day of the week, holidays and sinusoidal functions). Calendar variables are created based on the time index.

The learning model is a logistic regression giving the probability of occupancy. The dependent variable is the occupancy status (1 is occupancy, 0 is inoccupancy). The explicative variables are the calendar variables. The sinusoidal functions are used to obtain a smooth evolution of the probability of occupancy. The computing time of the learning model is less than a second.

3.3.2 The forecasting part

The learned model is used once to directly forecast the probability of occupancy for the next 48 hours with 5 minutes time step. Probabilities higher than 0.5 are transformed to 1 (occupancy) and probabilities lower than 0.5 transformed to 0 (inoccupancy).

3.3.3 Tool Result

The actual occupancy of the whole building^{xxv} vs occupancy predicted from the tool is displayed in graph below in Figure 16. Please note that the batch tool released as opensource produces result for next 24 hours with 5 minutes segmentation – below graph is part of the result zoomed to show occupancy during part of day time and at night.



Figure 16 Predicted (zone level) vs actual occupancy (overall sum) data result

Note: above data results is not cleaned to remove occupancy information. Ideally, data should be cleaned by removing all connections where zone name is empty.

Figure 17 shows the execution of this tool in real time (currently implemented as part of pilot3a) based on same data as used in this task. Since the tool executes in realtime and is continuously

running in Pilot3a, its actual data and prediction trend has been captured for one week. The graph shows data stream for predictions of day and night with a weekend in between.



Figure 17 Result of Occupancy Prediction with continuous execution for a period of one week

4 Conclusion

This deliverable reports the results of task 5.2 where we developed methods for integration and deploying PLATOON services to test their behavior based on components used in demonstration environment as well as the components made opensource from production environment. The deliverable explains the methodology adapted, a test use case of occupancy prediction to build the context for implementation identified from Pilot3a and implementation of data pipeline and the analytical tool to serve the sample use case. The methods presented in this deliverable are tested in T5.5 which is the task that focuses on the integration testing of components of the reference architecture. As a result of this task, the following major outcomes are obtained:

- For demonstration scenario, integration requirement is not too complex therefore components used may not have high availability and scalability requirement rather focus is on validating the architecture. The reference implementation is made available as opensource which integrates the following:
 - o Data pipeline is simple as raw data provided for demonstration is already aggregated from multiple sources and is provided in CSV format so there was no need of full ETL capability in semantic adapter;
 - o Semantic adapter used is based on SDM RDFizer which transforms data from CSV into RDF. Scalability of RDFizer is discussed in section 3.1.1.4;
 - o Federated query processing is used for query processing;
 - o A building occupancy tool with basic implementation; and
 - o Integration of dashboard for analytical tools results visualization.

The services presented in this scenario are tested in T5.5 which is the task that focuses on the integration testing of components of the reference architecture.

- In contrast for production scenarios, scalability is key, therefore, implementation is based on big data frameworks. Components involved in end-to-end service integration are made available as opensource as part of the PLATOON deliverables which integrates the following:
 - o Semantic adapter implementation is based on framework that is fully capable of building complex data ingestion pipelines involving a big data ETL process.
 - o A sample pipeline template for semantic adapter, SPARQL-Generate query to transform CSV data to RDF data.
 - o A building occupancy prediction tool based on deep learning model and trained with historical occupancy data is made open source. The tool is capable of executing on bigdata frameworks like Apache Flink and Apache Spark.

The services presented in this scenario are implemented in T6.1 pilot3a which is end to end implementation of complete pilot. However, the components presented in T5.2 that are made opensource can be used in demonstration scenario where availability/scalability requirements are larger than what the demonstration environment has to offer.

5 References

- ⁱ Scientific Data Management Group at TIB, “SDM-TIB/SDM-RDFizer,” 2022. [Online]. Available: <https://github.com/SDM-TIB/SDM-RDFizer>
- ⁱⁱ Maxime Lefrançois, Antoine Zimmermann, Noorani Bakerally A SPARQL extension for generating RDF from heterogeneous formats, In Proc. Extended Semantic Web Conference, ESWC, May 2017, Portoroz, Slovenia (long paper - PDF - BibTeX)
- ⁱⁱⁱ Platoon Semantic adapter. Available: <https://github.com/PLATOONProject/semantic-adapter>
- ^{iv} Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith. 2017. “Calibrating Noise to Sensitivity in Private Data Analysis”. Journal of Privacy and Confidentiality 7 Practical Privacy: The SuLQ Framework (Blum, Dwork, McSherry, and Nissim ‘05
- ^v Formalizing Tag-Based Metadata with the Brick Ontology. Gabe Fierro, Jason Koh, Shreyas Nagare, Xiaolin Zang, Yuvraj Agarwal, Rajesh K. Gupta, and David E. Culler. Frontiers in Built Environment, Vol 6 (September 2020). DOI: <https://doi.org/10.1145/3360322.3360862>
- ^{vi} C. F. Draschner, J. Lehmann and H. Jabeen, “DistSim - Scalable Distributed in-Memory Semantic Similarity Estimation for RDF Knowledge Graphs,” in *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, 2021
- ^{vii} Apache Spark™ - Unified Engine for Large-Scale Data Analytics. Available: <https://spark.apache.org/>
- ^{viii} Apache Jena - Available at: <https://jena.apache.org/>
- ^{ix} Apache Hadoop - Available at: <https://hadoop.apache.org/>
- ^x Sparqlify — Agile Knowledge Engineering and Semantic Web (AKSW) (no date). Available at: <https://aksw.org/Projects/Sparqlify.html>
- ^{xi} F. Bakhshandegan Moghaddam, C. Draschner, J. Lehmann and H. Jabeen, “Literal2Feature: An Automatic Scalable RDF Graph Feature Extractor,” in *Proceedings of the 17th International Conference on Semantic Systems, SEMANTICS 2021*, 2021.
- ^{xii} C. F. Draschner, C. Stadler, F. Bakhshandegan Moghaddam, J. Lehmann and H. Jabeen, “DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.
- ^{xiii} F. Bakhshandegan Moghaddam, C. F. Draschner, J. Lehmann and H. Jabeen, “Semantic Web Analysis with Flavor of Micro-Services,” in *LAMBDA Big Data Analytics Summer School. Doctoral Workshop*, 2021 forthcoming
- ^{xv} R2RML: RDB to RDF Mapping Language.” Available: <https://www.w3.org/TR/r2rml/>.
- ^{xvi} <https://rml.io/specs/rml/>
- ^{xvii} Semantic adapter (<https://github.com/PLATOONProject/semantic-adapter>)
- ^{xviii} SparqlGenerate was contributed and sponsored by Engie as part of SEAS project. For more information about SPARQL generate <https://ci.mines-stetienne.fr/sparql-generate>
- ^{xx} Olivier Corby, Catherine Faron Zucker, Fabien Gandon. LDScript: a Linked Data Script Language. [Research Report] RR-8982, INRIA. 2016. (hal-01402901)

^{xxii} GitHub. “PLATOONProject/Occupancy-Semantic-Pipeline: As a Part of T5.2 Deliverable, Repository Contains Occupancy Semantic Adapter Flow and SparqlGenerate to Semantify Data as per Pilot3a Data Model.” <https://github.com/PLATOONProject/occupancy-semantic-pipeline>.

^{xxiii} ‘SANSa-Stack – Scalable Semantic Analytics Stack’ (no date). Available at:

<https://sansa-stack.net/>

^{xxiv} *PLATOONProject/pilot3.occupancy-prediction.batch: Building occupancy prediction tool* (no date) *GitHub*. Available at: <https://github.com/PLATOONProject/pilot3.occupancy-prediction.batch>

^{xxv} Pilot3a data is from Engie Lab office building located at 4 Rue Joséphine Baker, 93240 Stains FRANCE