D4.7 Semantic interoperable middleware - final version

digital-water.city
digitalwater_eu

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Deliverable N°4.6	Semantic interoperable middleware
Related Work Package	WP4
Deliverable lead	21-SINTEF
Author(s)	Bård Johan Hanssen
	Peter Haro
	Audun Vennesland
Contact for queries	bard.hanssen@sintef.no
Grant Agreement Number	n° 820954
Instrument	HORIZON 2020
Start date of the project	01 June 2019
Duration of the project	42 months
Website	www.digital-water.city
License	This work is licensed under a Creative Commons Attribution 4.0 International License
Abstract	D4.7 describes and outlines the semantic interoperable middleware developed for the DWC project. It introduces the concepts and components that are utilized in order to build the middleware and discusses how these are used in the Paris case in DWC. It also outlines the process and smart-data input to the SAREF ontologies. This document will also detail the implementation and usage of the middleware and possible future opportunities within the DWC project consortia.

Dissemination level of the document

X	PU	Public
	PP	Restricted to other programme participants
	RE	Restricted to a group specified by the consortium
	CO	Confidential, only for members of the consortium

Versioning and contribution history

Version*	Date	Modified by	Modification reasons
D1	23.11.2020	Peter Haro	First draft of document deliverable
D2	27.11.2020	Bård Johan Hanssen	Sent draft version to reviewers
R1	08.12.2020	Peter Haro	Changed document according to reviews
R2	08.12.2020	Nico Caradot	Review by coordinator
S1	08.12.2020	Peter Haro	Final version for submission
D3	15.06.2022	Peter Haro	Sent draft of final deliverable for review
R3	29.06.2022	Bård Johan Hanssen	Changed document based on review feedback
D4	18.07.2022	Peter Haro	Final version for submission
S2	15.08.2022	Nico Caradot	Final check before submission
R4	03.01.2023	Audun Vennesland	Added section 3.3 addressing comments from external review
V	13.01.2023	Peter Haro	Final revision for final submission

* The version convention of the deliverables is described in the Project Management Handbook (D7.1). *D* for draft, *R* for draft following internal review, *S* for submitted to the EC (under external review) and *V* for approved by the EC. Note that previous version to *V* are draft since they are not yet approved by the EC.

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Table of content

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

List of figures

List of tables

Glossary

Term	Description
API	Application Programming Interface
CAP	Operator of Greater Milan wastewater treatment
CoP	Community of Practice
DWC	Digital Water City
Environment system	The environment (system) includes everything that is not a part of the target system, and which interfaces the target system directly. This includes both stakeholders and other systems.
EWS-BWQ	Early Warning System for Bathing Water Quality to be developed in Paris.
EWS-SWR	Early Warning System for Safe Water Reuse to be developed in Milan.
FIB	Fecal Indicator Bacteria
FIWARE	FIWARE is an initiative that defines reusable open-source components and standardised specifications for context data management. https://www.fiware.org/
JSON	JavaScript Object Notation. A lightweight data interchange format.
KWB	Kompetenzzentrum Wasser Berlin

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

NGSI	Next Generation Service Interface. A protocol to manage context information.
OWL	Web Ontology Language. A Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things.
IoT	The Internet of Things – The description of a network of physical 'things' whos also connected and exchanging data over a network or internet.
HTTP	Hypertext transfer protocol: the standard protocol for transferring hypertext documents on the World Wide Web.
RDF	Describes data by defining relationships between data entities expressed using URIs (Uniform Resource Identifiers) and related via triples in the form (subject-predicate-object).
CQRS	Command Query Responsibility Segregation - Separates read and update operations for a data store
SOTA	State Of The Art
REST	Representational State Transfer. A software architectural style that defines constraints to be used for creating web services.
SIAAP	Syndicat Interdepartmental pour L'Assainissement de l'Agglomeration Parisienne, Operator of greater Paris wastewater treatment
SPARQL	A query language that can be used to express queries across diverse data structures following the RDF specification.
Target system	The system for which the architectural description is created. This includes both stakeholders (human, organisation) and systems.
UAV	Unmanned Aerial Vehicle
YAML	YAML Ain't Markup Language - YAML is a human friendly data serialization standard for all programming languages.
UML	Unified Modelling Language, a modelling language for describing system- and software architecture.
View	A representation of a whole system from the perspective of a related set of concerns.
Viewpoint	A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purpose and audience for a view and the techniques for its creation and analysis.
WWTP	WasteWater Treatment Plant
XML	eXtensible Markup Language. A markup language that defines a set of rules for encoding, storing and communicating data.

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Executive summary

Deliverable (D) 4.7 is the fifth out of five deliverables, produced by Work Package (WP) 4 of the DWC projects, which pertains to managing data and systems in an interoperable way. This document builds upon D4.4 which details the semantic interoperability design requirements and D4.1, which outlined an interoperable and secure flow of information. Within the water sector, there exist a myriad of under- or un-utilized data within both the consortium and the domain in total. Therefore, we proposed the development of a middleware framework consisting of generic components for semantic alignment and matching to be able to utilize existing and novel data, across multiple systems and software components.

This report describes the development of a microservice structured middleware that provides data transformation from different file formats such as CSV, TSV, JSON into the FIWARE NGSIv2[1] format across multiple protocols, allowing stakeholders to easily integrate their systems and data sources into the DWC FIWARE ecosystem without having to write their own data converters. The middleware has been implemented following the requirements described in D4.4 [1]. While the original idea of developing a microservice framework with generic components was changed to utilizing FIWARE instead, the implementation is still using a microservice architecture albeit as a part working in tandem with FIWARE and exposes an OpenAPI[2] endpoint for users.

Following the implementation details, the report also describes how the middleware is applied in one of the project pilots in the Paris Case Study, which monitors bathing water quality using data from multiple data sources, detailing how the middleware handles data sources with different access levels and authorizations.

The report also describes the development and changes to the smart data models used in the project, as well as the proposed inputs to the existing SAREF ontologies, and the process for doing so.

[1] https://fiware.github.io/specifications/ngsiv2/stable/

[2] https://swagger.io/specification/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

1. Introduction

1.1. Summary of DWC and objectives of work package 4 and subtask 4.3.3

DWC aims at creating digital solutions to link water management in the physical world to the digital spheres such as sensor networks, real-time monitoring, machine learning, etc.

24 partners from 10 countries work together in the case study cities Berlin, Copenhagen, Milan, Paris and Sofia and support the utilities and municipalities in improving water quality, return on investment and public information about water-related issues.

Core of work package (WP) 4 is to ensure that the digital solutions are designed, developed and deployed in a way in which data and information are exchanged with the utility systems in a cyber-secure and interoperable way. This includes a risk analysis and proposition of risk reduction measures to protect data and systems from unauthorized access as well as the description of semantic models and interoperability design guidelines. The tasks focus on the digital solutions and their impacts on the existing infrastructures, but not on general cyber-physical risks.

As the DWC project progressed, the nature of subtask 4.3.3 of WP4 changed. The subtask was originally aiming to create a reusable microservice framework providing generic components for communication, data management and semantic interoperability. However, in the early stages of the Digital-Water.city project these objectives changed and it was decided at project management level that the existing FIWARE framework should be used instead. This decision was primarily made based on that other collaborating research projects in the DigitalWater 2020 and ICT4Water clusters were also implementing digital solutions based on FIWARE and FIWARE already offered an architecture and a set of open-source components that could support interoperability in the sector. Furthermore, multiple stakeholders in the project identified use cases, which aligned with FIWARE. Thus, 4.3.3 has changed character where we provide pluggable components interoperable with the FIWARE ecosystem as well as creating data transformers for the standardized data models and ontologies create in 4.3.1 and 4.3.2 as well as contribute to develop standardized data models and ontologies for the water management domain to facilitate semantic interoperability. These components can be placed within existing data pipelines and FIWARE instances as middleware. As a result, the original development on the framework was stopped in favor of this solution, and development was aligned with subtask 4.3.1 (Semantic interoperability design requirements) and 4.3.2 (Developing the DWC reference ontology) and how the partners are utilizing FIWARE, with an initial focus on the implementations created in Milan and Paris.

The subtasks 4.3.1 and 4.3.2 have mapped out several use-case scenarios for the digital solutions in the DWC project where they will require some middleware to perform semantic interoperability. These use-case scenarios are described in D4.1 [2], D4.4 [1] and D4.5 [3]. Due to the changing requirements of the middleware described above, we decided to focus on some specific use cases instead of focus on generic functionality. Thus, we have in this work focused on mapping requirements related to the physical and logical architecture of the Paris case so that we can better support interoperability through the means of standardized data models and middleware components. We focused on the Paris case as it brought up interesting challenges in aspects regarding data ingestion, interoperability and use. Therefore, the output from these subtasks is a key driver for this subtask providing both the design requirements but also providing key insights into the existing solutions and how we can interact with those. With collaboration with the partners in 4.3.3 we have also received feedback on our solution and how they wish to utilize it to achieve semantic interoperability across their solutions.

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

As part of the development of the semantically interoperable middleware the work described in this report targets three objectives: (1) introduce the software components that will be used in order to create the semantic middleware, including FIWARE and the surrounding architecture; (2) document how the development of the middleware component(s) have been done so far, as well as the future plans for development; (3) document the requirements as generically as possible in order to support reusability in other deployments taking place in the DWC project. With respect to the latter objective, the structuring and notation used in this report is guided by the ARCADE Framework[3], a domain- and technology independent architectural description framework for software intensive systems.

1.2. Structure of this document

D4.7 consists of 5 main chapters, starting with the overall approach in chapter 2, describing the link between other deliverables and an overview outlining the core technologies and general ICT components utilized to realize development of the DWC middleware.

Chapter 3 outlines the development process and software architecture for the DWC semantic software solution, how it interacts with FIWARE and how it can be used by the DWC solutions. It gives a brief overview of containerization, how it's used in D4.7 and how it enables us to create coordinating components. Furthermore, it details the core context management in FIWARE and how it can interact with other systems. Chapter 4 subsequently details its usage in the Paris case and its interaction across multiple systems by multiple vendors. It also shows both technical and semantic interoperability between systems, services, and data.

Chapter 5 describes how D4.7 assists in standardizing data models and how it extends and creates new data models for DWC. It details the procedures for submitting requirements to smartdatamodels.org and the extensions for those models for use in DWC. Whereas chapter 6 details our input to the SAREF ontologies and proposed extensions from the DWC project.

New in this report from the previous D4.6 report is the more detailed description of how the middleware solution has been implemented, as many of the future developments mentioned in D4.6 have now been implemented. Additionally, it details the changes made to the Smart Data Models since the previous deliverable as well as the change requests made towards the SAREF ontologies used in DWC.

Following external review, section 3.1 has been clarified and a new section 3.3 has been added to provide details on how the semantic middleware actually uses OWL data models.

2. Overall Approach

We decided to create novel components working in tandem with the FIWARE ecosystem based on the sketches outlined in the 4.4 deliverable. D4.4 [1] describes design requirements for implementing a semantic interoperability middleware architecture and will outline how we implemented these requirements.

[3] http://arcade-framework.org/assets/documents/ARCADE-Example.pdf

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

We will follow the development processes as outlined utilizing a microservice architecture where we compose our components into reusable containers, based on Docker[4]. This chapter begins with a brief introduction to the most relevant concepts from both Docker and the FIWARE framework before we denote how we will move all developed code from the existing microservice framework into the ecosystem.

2.1. Link between D4.4 and D4.7

The requirements captured in D4.4 [1] will serve as the basis for the development of the semantic interoperability software outlined in this document. It serves as a classic requirement engineering process where requirement elicitation is the first natural step. This basis will allow us to create a more complete ontology for the DWC project as we will be able to design a more semantically relevant model. However, some information is also backported as the initial development performed in D4.6 [4] could be showcased for the partners in meetings w.r.t the virtual workshops performed in D4.4 [1] so that we could collect feedback as early as possible.

The decision to create a containerized application that works either within FIWARE or as a standalone component(s) also comes from the elicitation process performed in D4.4 [1] as not all partners fully understand how to utilize FIWARE and elect to perform processes more familiar to them. Therefore, the process is performed in iterations where we continuously seek to gather requirements as we develop so that we can change the software across releases until the final delivery.

2.2. Introduction to Containerization

Containerization is an approach to software development in which an application or service, its dependencies, and its configuration (abstracted as deployment manifest files) are packaged together as a container image. The containerized application can be tested as a unit and deployed as a container image instance to the host Operating System (OS)[5].

Software containerization enables developers and IT professionals to deploy them across environments with little or no modification. Importantly, this also allows them to isolate applications from each other on a shared OS, cloud system or with segregated networks. Containerized applications run on top of a container host that in turn runs on the OS (Linux or Windows). Containers therefore have a significantly smaller footprint than classical virtual machine (VM) images.

Another benefit of containerization is scalability. You can scale out quickly by creating new containers for short-term tasks. From an application point of view, instantiating an image (creating a container) is like instantiating a process like a service or web app. This is also very beneficial for reliability, as it is quick and easy to spin up a new container should one crash, and they can be run in parallel on different hosts or VM, across domains. In short, containers offer the benefits of isolation, portability, agility, scalability, and control across the whole application lifecycle workflow. The most important benefit is the environment's isolation provided between Dev and Ops.

[4] https://www.docker.com/

[5] https://aka.ms/microservicesebook

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

2.3. Introduction to Docker

Docker is an open-source project for automating the deployment of applications as portable, self-sufficient containers that can run on the cloud or on-premises. To support this functionality Docker provides a generic platform on demand for running these containers. Docker containers differentiate themselves from Virtual Machines because it uses only OS-level virtualization to run a container, without the developer having to think about the underlying platform. This means that multiple containers run on only a single OS kernel.

Therefore, Docker containers can run anywhere, on-premises in the customer datacenter, in an external service provider, on a research lab or local machine, or in the cloud. Developers can use development environments whether on Windows, Linux, or MacOS as such making Docker suitable in interdisciplinary projects with cross-cutting concerns.

2.4. Introduction to FIWARE

FIWARE is a framework of open-sourced software components targeted towards digitalisation and smart application of data across multiple application domains. The focal point of FIWARE is interoperable solutions for context management. This includes the ability to source data from measurement devices (e.g., sensors), represent these source data in a wider context representation, and provide the means for accessing these context data by end-user applications. The five architectural perspectives of the framework are illustrated in Figure 1. In the following sections, we will focus on the three in the middle, namely Core Context Management, Interface with IoT, and Context Processing, Analysis and Visualisation, as these are the ones that are most relevant for the middleware being developed.

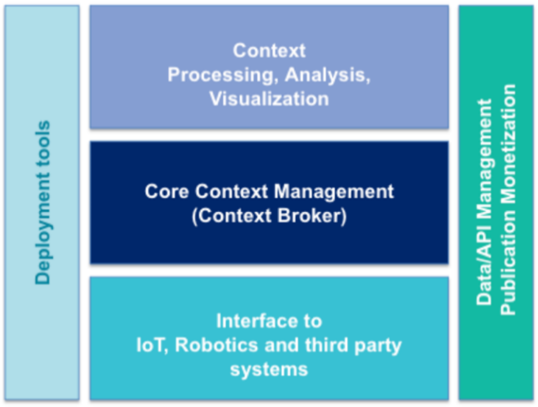

Deployment tools
Context Processing, Analysis, Visualization
Core Context Management (Context Broker)
Interface to IoT, Robotics and third party systems
Data/API Management Publication Monetization

Figure 1. FIWARE Framework[6]

[6] Illustration taken from https://www.fiware.org/developers/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

2.4.1. Core Context Management

The Core Context Management part of FIWARE represent the ability to produce, gather, publish and consume context data and turn this into actionable information to be applied by end user applications.

The context data are represented through values assigned to attributes that contribute to define the entities and are managed by a *Context Broker*. A Context Broker is a core and mandatory component of the FIWARE framework that allows for storing, updating and subscribing to the entities representing the context via a standardised REST API (NGSIv2 or NGSI-LD as described below). Orion[7] is a Context Broker that has been released by FIWARE. Orion provides an NGSI v2 API. In addition, the following Context Brokers providing NGSI-LD APIs are under incubation: Orion-LD Context Broker[8], the Scorpio Broker[9], and the Stellio Context Broker[10].

In addition to the Context Broker the Core Context Management also include Generic Enablers[11] that enable to store context data persistently, such as STH-Comet[12], Cygnus[13], Draco[14] and QuantumLeap[15].

The context itself is defined by means of the FIWARE NGSI[16] API. NGSI defines a data model for describing context information; a context data interface for exchanging information via queries, subscriptions, and updates; and a context availability interface for exchanging information on how to obtain context information. There are basically two NGSI versions that are relevant, NGSIv2 and NGSI-LD. The main elements in the NGSIv2 model are entities, attributes, and metadata, as shown in Figure 2. An entity represents a physical or logical object (e.g., a sensor, a person, an issue in a ticketing system) defined by an identifier and a type definition, an attribute represents some property of the entity (e.g., a measurement value), while the metadata describes additional “data about the data”, such as the accuracy of the measurement value or the location of a sensor.

7 https://fiware-orion.readthedocs.io/en/master/#welcome-to-orion-context-broker

8 https://github.com/FIWARE/context.Orion-LD

9 https://github.com/ScorpioBroker/ScorpioBroker

10 https://github.com/stellio-hub/stellio-context-broker

11 A Generic Enabler is a component that is considered general purpose and independent of any particular usage area.

12 https://github.com/telefonicaid/fiware-sth-comet

13 https://fiware-cygnus.readthedocs.io/en/latest/

14 https://github.com/ging/fiware-draco

15 https://github.com/smartsdk/ngsi-timeseries-api

16 https://fiware.github.io/specifications/ngsiv2/stable/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

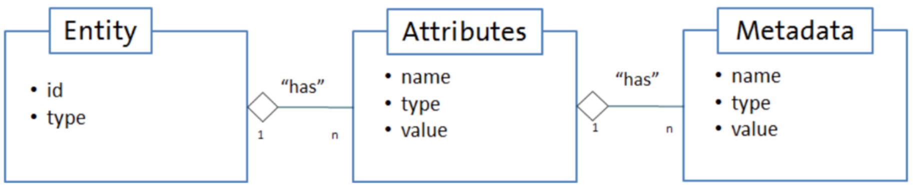


Figure 2. NGSIv2[17]

The other version of NGSI, NGSI-LD, where LD stands for Linked Data, has a different underlying data model than NGSIv2. Here, data are described in triples in a subject-predicate-object pattern resulting in a graph representation of the context data. Furthermore, in NGSI-LD the ID of an entity should be a Uniform Resource Identifier (URI) ensuring a consistent representation of the identifier of an entity.

Figure 3 shows the underlying model of NGSI-LD. As the figure shows, there is no metadata element, and the Attributes element in NGSIv2 now refers to either Property or Relationship where the former represents literal values (strings, decimals, etc.) while the latter represent relationships between different entities.

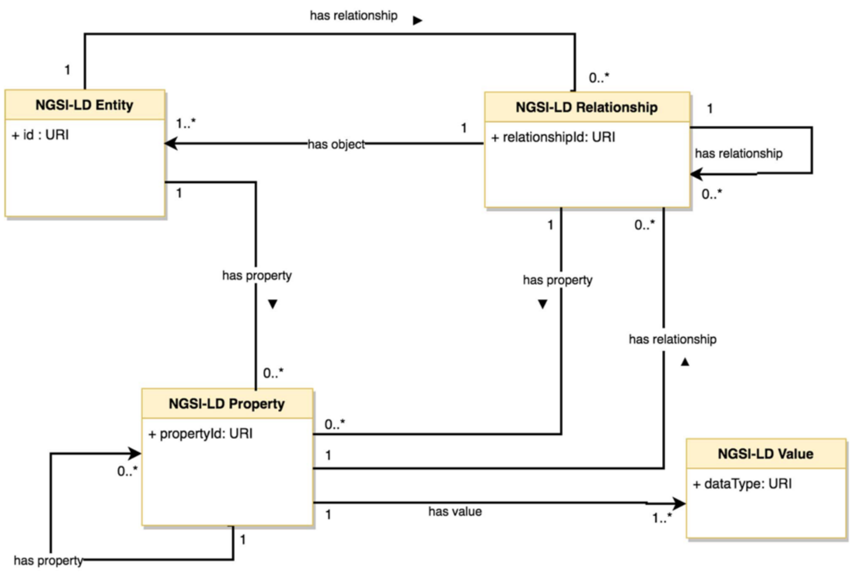


Figure 3. NGSI-LD[18]

[17] Illustration taken from https://www.fiware.org/developers/

[18] Illustration taken from https://www.fiware.org/developers/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Figure 4 and Figure 5 illustrate how data are formatted using NGSI v2[19] and NGSI-LD[20]. These JSON snippets, which both represent an extract of the WeatherObserved data model[21], shows how temperature and precipitation is represented along with a timestamp and information about the weather station providing the measurements.

As these examples show, there are some notable differences in how data are represented. First, NGSIv2 is represented as basic JSON format[22], while NGSI-LD is represented using JSON-LD[23]. Further, in NGSI-LD the ID shall be represented using a URI, not a simple string value as in NGSIv2. Each attribute in NGSI-LD shall contain two fields, a property and a value, whereas in NGSIv2 an attribute can be represented by just a value. In NGSI-LD, a context element is added to provide fully qualified names (URIs) associated to terms. This is like how namespaces are used in XML.

[19] https://fiware.github.io/specifications/ngsiv2/stable/

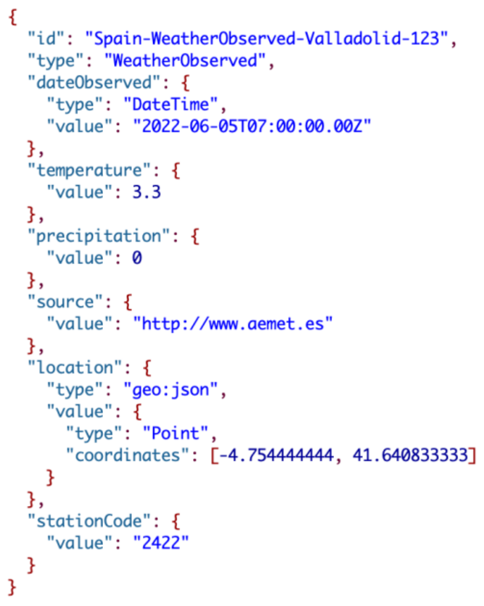
[20] https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf

[21]See https://github.com/smart-data-models/dataModel.Weather/blob/master/WeatherObserved/README.md

[22] https://www.json.org/json-en.html

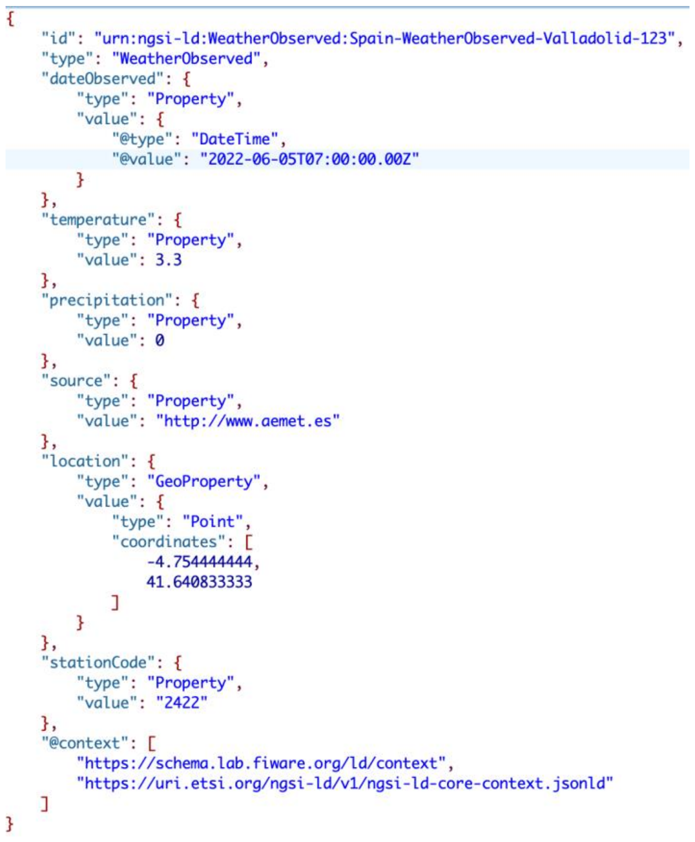
[23] https://json-ld.org/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.



```
{
  "id": "Spain-WeatherObserved-Valladolid-123",
  "type": "WeatherObserved",
  "dateObserved": {
    "type": "DateTime",
    "value": "2022-06-05T07:00:00.00Z"
  },
  "temperature": {
    "value": 3.3
  },
  "precipitation": {
    "value": 0
  },
  "source": {
    "value": "http://www.aemet.es"
  },
  "location": {
    "type": "geo:json",
    "value": {
      "type": "Point",
      "coordinates": [-4.754444444, 41.640833333]
    }
  },
  "stationCode": {
    "value": "2422"
  }
}
```

Figure 4. NGSIv2 example



```
{
    "id": "urn:ngsi-ld:WeatherObserved:Spain-WeatherObserved-Valladolid-123",
    "type": "WeatherObserved",
    "dateObserved": {
        "type": "Property",
        "value": {
            "@type": "DateTime",
            "@value": "2022-06-05T07:00:00.00Z"
        }
    },
    "temperature": {
        "type": "Property",
        "value": 3.3
    },
    "precipitation": {
        "type": "Property",
        "value": 0
    },
    "source": {
        "type": "Property",
        "value": "http://www.aemet.es"
    },
    "location": {
        "type": "GeoProperty",
        "value": {
            "type": "Point",
            "coordinates": [
                -4.754444444,
                41.640833333
            ]
        }
    },
    "stationCode": {
        "type": "Property",
        "value": "2422"
    },
    "@context": [
        "https://schema.lab.fiware.org/ld/context",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
    ]
}
```

Figure 5. NGSI-LD example

2.4.2. Interface with IoT

In order to interface with devices and systems providing context data to the Context Broker and its NGSI API, the IDAS Generic Enabler provides a set of IoT Agents supporting different IoT protocols. Currently IoT Agents for the following protocols are provided: JSON (over HTTP/MQTT), Lightweight M2M (LWM2M), Ultralight, LoRaWAN, OPC-UA and Sigfox.

2.4.3. Context Processing, Analysis and Visualisation

This architectural layer provides Generic Enablers that aim to enable processing, analysis and visualisation of context information. Examples of Generic Enablers in this area are: Wirecloud[24] for visualisation of integrated data, Knowage[25] for business analytics and Kurento[26] for real-time processing of media streams.

[24] https://wirecloud.readthedocs.io/en/stable/

[25] https://knowage.readthedocs.io/en/latest/

[26] https://kurento.readthedocs.io/en/stable/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

2.5. Introduction to the DWC semantic interoperability development methodology

The DWC digital solutions are developed by various stakeholders with differing experience with regards to DevOps. The IT/OT operations are placing their primary importance of adamant security, ensuring that additional software being installed does not weaken or expose the security of the existing solutions, as well as to provide robust software, which the different teams, with their different constrains can use. We decided to utilize containerization to support their production operations, so that the IT/OT team(s) can decide within which network segregation they want the software deployed. Ensuring that the local experts have control over their software will allow them to configure only the parts they require to reduce the attack surface.

We originally planned to use a microservice oriented architecture which would orchestrate and coordinate multiple services, interfacing with multiple types of databases while supporting different communication protocols. To fulfil these requirements in a generic manner each microservices supported CQRS communicating primarily using either HTTP(s) or AMQP as between client apps and the microservices, using asynchronous communication for data updates. These were propagated based on Integration Events and an Event Bus. However, since FIWARE supports this type of architecture we changed our building blocks to an interoperable container based on HTTP communication which can leverage the DRACO broker from FIWARE for intersystem communication like a distributed event bus, i.e., a distributed software bus that interconnects multiple disparate systems.

The development of the semantic interoperability software has used a design science approach based on a framework developed for R&D and the development of information systems [5]. The approach facilitates an agile development process, which focuses on user-centric design, early validation and reusing existing software. This process is especially applicable in multidisciplinary projects like DWC. The following methods have been selected to ensure close collaboration with industry and that the results are planned for usage in active operations:

- both the design and development processes were conducted in close collaboration between all actors involved in the system so that we could quickly start prototyping architecture and logical units.
- qualitative studies of the various prototypes developed throughout the project was evaluated with multiple facets so that we could understand user-interactions, the ramifications to their existing infrastructure and the movement throughout the supply chains. The project adopted multiple iterations using an agile approach. The semantic interoperability prototype was developed based on SOTA open-source frameworks, e.g., OWL. We have enhanced it and added missing features of the platform.

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

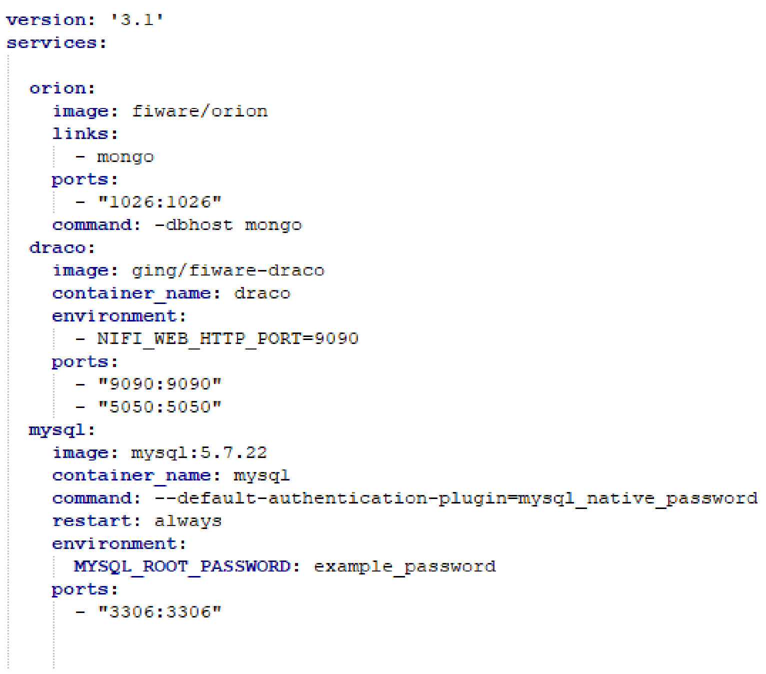
3. Utilization and development of interoperability software

The semantic interoperability software has been developed as container(s) that can run either inside the FIWARE ecosystem or as a stand-alone component translating and matching resources using either FIWARE or SPARQL or another generic linked datastore. We do this to support a standalone application while being interoperable with the future semantic work ongoing in the FIWARE project and SAREF4WATR[27].

3.1. Initial development

During the initial developments on the interoperability software, the FIWARE components and other applications that were to be used were tested and deployed as separate Docker images. Building upon this, the "dockerized" FIWARE components were then added to a compose file which could then be used to set up and deploy all included images without any additional input, excluding any local configurations like changing passwords and which ports to use.

Figure 6 shows an example of a minimal docker compose file[28], which is a YAML file defining services, networks, and volumes, to set up docker images for a FIWARE Orion Context Broker and FIWARE Draco Generic Enabler as well as their dependencies. Both these are compliant with NGSI v2. Note that while ports and passwords are hard coded in this example, they can just as easily be read from a config file or passed as environment variables at runtime.



```yaml
version: '3.1'
services:

  orion:
    image: fiware/orion
    links:
      - mongo
    ports:
      - "1026:1026"
    command: -dbhost mongo
  draco:
    image: ging/fiware-draco
    container_name: draco
    environment:
      - NIFI_WEB_HTTP_PORT=9090
    ports:
      - "9090:9090"
      - "5050:5050"
  mysql:
    image: mysql:5.7.22
    container_name: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: example_password
    ports:
      - "3306:3306"
```

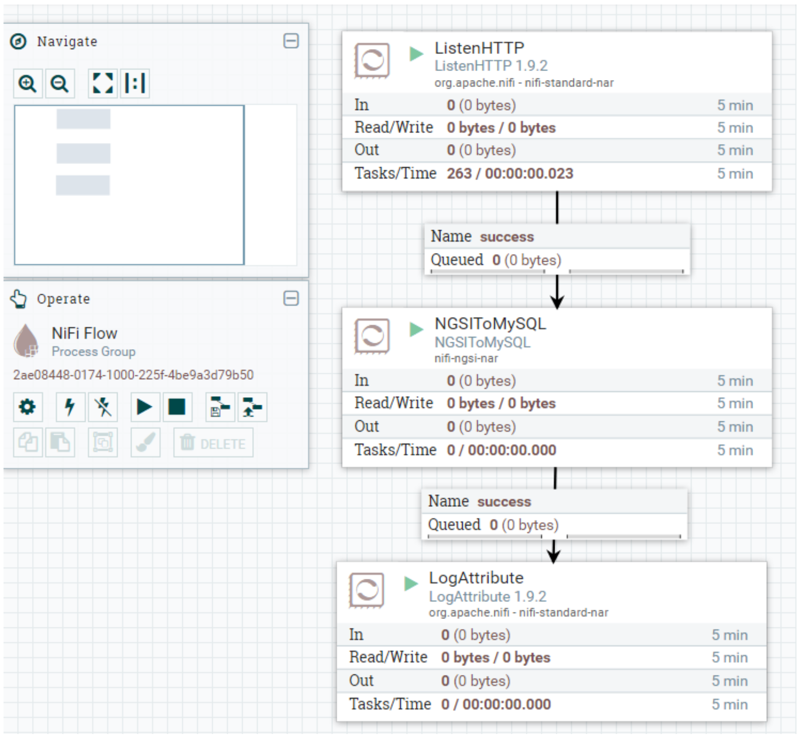
Figure 6. An example compose file for a Generic Enabler and Context Broker

[27] https://saref.etsi.org/saref4watr/v1.1.1/

[28] https://docs.docker.com/compose/compose-file/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Since a Context Broker does not store historical data, it is useful to set up third-party storage of some kind. Figure 7 shows how a FIWARE Generic Enabler can be set up to subscribe to the data stream of a Generic Enabler and store updates values to a MySQL database. This is done using a set of processors that handle the tasks required to write the data to more persistent storage. First, the initial processor establishes an HTTP connection and sets up a subscription in the Context Broker, specifying which events it should be notified of. The second processor handles the task of reading the new data and storing it to the database. While not required, the third processor does the work of logging the actions of the other processors if configured to do so.



ListenHTTP
ListenHTTP 1.9.2
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	263 / 00:00:00.023	5 min

Name success
Queued 0 (0 bytes)

NGSIToMySQL
NGSIToMySQL
nifi-ngsi-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 0 (0 bytes)

LogAttribute
LogAttribute 1.9.2
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Figure 7. FIWARE Draco data flow from Orion to MySQL

3.2. Data management middleware

To facilitate all digital solutions in the DWC project with a semantic interoperability software, we needed to ensure that we developed something that was applicable to the implementors. As the development must work in conjunction with FIWARE without the partner developers having to adapt their solutions to integrate the software, we decided to deliver the software as standalone components. Therefore, we leverage both the DRACO generic enabler and the Origin broker as described in 3.1 and 2.4.

As there are multiple levels of network segregation between the different IT/OT teams in the project consortia, they use a multitude of protocols and methods for publishing data from their SCADA or other types of secure systems. Although HTTP is a close to ubiquitous protocol, multiple partners utilize other methods for data transfer, especially in push configurations, meaning that data is pushed from the secure system(s) to FIWARE through our middleware. Furthermore, the systems work on differing

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

logical levels, such as on file systems and FTP/SFTP/SCP using files as transmission medium. To support the different modes of operations we've created a data management component which acts as forwarder and conversion agent to the other FIWARE components.

This system implements a physical filesystem provider, working on both Windows, Linux, Samba, NTFS, and UNC to monitor a filesystem hierarchy detecting new, changed, deleted, or appended files and will perform necessary file conversion(s) to the data which support semantic data in FIWARE. Thus, allowing the various stakeholders to interact with FIWARE without changing existing workflows by converting the different files and datums into NGSI v2 so that the other components can interact with semantically enriched data.

Using the methodology outlined in 3.1 for data storage in combination with traditional data ingestion, either through HTTP/MQTT or other IoT protocols, allows us to combine the persistent data with semantic models, linked data, and ontologies. This means we can use the stored sensor- and operational data in combination the semantic data structures, both static and dynamic. This is beneficial because it's impossible to create an all-inclusive ontology, therefore we use local domain ontologies and databases which allows us to create new ontology elements automatically using a reasoner [6]. This process is what enables us to create a standalone microservice, which can also be used within the FIWARE ecosystem. Using a simple API gateway or an orchestrator allows the application to be "context-independent" i.e. it uses HTTP with a configured persistent storage (such as configured in the compose file), to provide both intra- and intercommunication.

There were two primary reasons for selecting this method. The first being that it would allow us to place the container within or outside of the FIWARE ecosystem while still supporting both systems and it allows industrial partners or other stakeholders to utilize private datasets, which can be, but does not have to be shared in a global network. Utilizing a combination of public and private data enables us to provide more sophisticated matchmaking in conjunction with the Ontology, which can result in better data quality. The increased quality comes from having more instance data in the ontology, which allows us to do interference and exchange data more accurately between the entities registered, thus increasing the quality of the middleware. For new entities that are not registered in the ontology, we can possibly infer its property and perform matching by utilizing a reasoner based on the ontology and existing instance data. This means we can better match resources (registered entities in the software i.e., IoT-sensor) based on their semantic properties due to having access to more data.

The second reason was that for many operations, the stakeholders IT/OT team(s) do not want to manage any more dependencies than necessary. Deploying the container as a FIWARE component allows the software to be managed as a part of their existing FIWARE management, reducing the overhead for operations. Although, for stakeholders who do not need FIWARE and simply interacts with it, a standalone module will provide less configuration and management overhead. This allows us to interact with multiple data stores without necessarily having permission access to the same networks or ecosystem.

3.3. Extending the middleware with additional semantic functionality

In complex and dynamic data exchange scenarios, multiple data providers and consumers may be involved in the data exchange, and the data integration process will likely have to deal with complex data formats, possibly describing data from other domains but water management. The goal of the semantic interoperability middleware is to employ semantics to support mediation between different data providers/consumers and a FIWARE Context Broker. Here, semantics refer to the DWC Reference Ontology and techniques for inferring the underlying meaning of concepts in order to map different formats. Preparing for such data integration in a fully manual fashion will require significant resources when mapping the syntax and semantics of these different data exchange, hence, semi-automated methods should be explored. The system architecture of the semantic interoperability middleware prepares for such semi-automated data integration scenarios. Here, different types of syntactic and semantic matching algorithms can be applied to identify mappings between data exchange formats to the DWC Reference Ontology (see D4.5 [3] for more details), effectively supporting ontology-based data integration (OBDI) [7]. The mapping between the ontology concept (to which elements in the different source data schemas are mapped to) and the corresponding element in the appropriate standardised data model used to store data in the FIWARE Context Broker is supported by the context references provided by the Smart Data Models initiative[29].

3.3.1. Overall systems architecture for the Semantic Interoperability Middleware

Figure 8 depicts the core software components needed to enhance the middleware as described above. The DWC Reference Ontology will be used as a mediator between the formats used by data consumers / providers interacting with the FIWARE Context Broker. The Syntactic Matching Services and the Semantic Matching Services implemented as micro services will employ techniques from the research areas of schema- and ontology matching [8] to match elements of the data exchange formats to concepts defined in the DWC Reference Ontology.

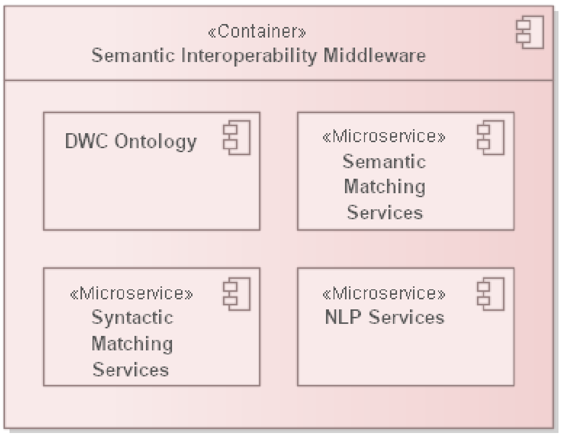

«Container»
Semantic Interoperability Middleware
DWC Ontology
«Microservice» Semantic Matching Services
«Microservice» Syntactic Matching Services
«Microservice» NLP Services

Figure 8. Overall component view of Semantic Interoperability Middleware.

From the DWC Reference Ontology these techniques can employ both concept names, associated properties (and hence structural characteristics) and textual annotations associated with both concepts and properties. Here, we define Syntactic Matching Services as basic string-matching techniques, such as variants of Edit distance[30], while Semantic Matching Services use techniques that aim to capture the underlying meaning of words and identify similarity despite differences in word compositions. In the latter category of techniques, we also include external sources of background

[29] https://github.com/smart-data-models/data-models/tree/master/context

[30] https://en.wikipedia.org/wiki/Edit_distance

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

knowledge, such as WordNet[31] and BabelNet[32]. Natural Language Processing (NLP) services, such as such as those included in Laser Translation Toolkit[33], can support the Semantic and Syntactic Matching Services by e.g., parsing natural language definitions associated with the data exchange formats and concepts in the DWC Reference Ontology. The value of such an approach will self-reinforce as the number of element-to-concept mappings increase. In this case the element-to-concept mappings can be employed in the matching process.

Figure 9 shows a high-level sequence view involving the middleware. Here, data exchange formats used by the Target System (i.e., the FIWARE Context Broker in our case) and the Source System (i.e., the system that will provide / consume data to/from the FIWARE Context Broker) are mapped to the DWC Reference Ontology. The mappings, containing element-wise alignment between the data exchange formats and the DWC Reference Ontology, are returned to the respective systems and stored within the middleware. Of course, the automatically generated mappings must be verified by human interaction before they are stored within the middleware. This way, whenever a Source System will interact with the FIWARE Context Broker, it may continue to use its original data exchange format.

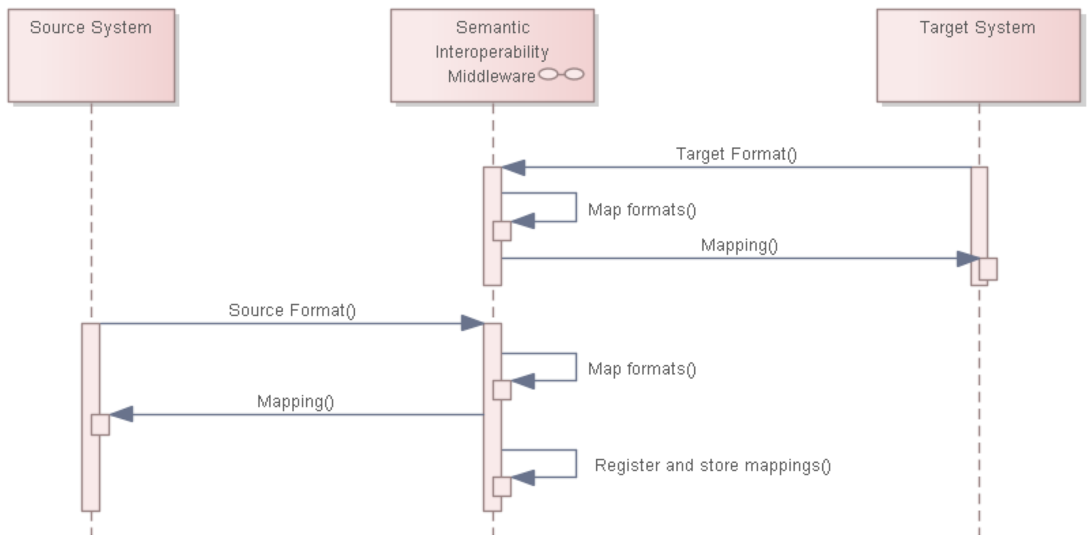


Figure 9. Overall sequence view of Semantic Interoperability Middleware.

The benefit of using such an approach is when there are multiple Source Systems using different data exchange formats that map to a single ontology or multiple ontologies.

3.4. Future developments

As the system matures and more instance data is added to the system the semantic middleware can benefit from combining its reasoning capabilities with machine learning on the semantically enriched data for a hybrid system. One of the main advantages of the ontologies created for DWC is the ability to add new data to the system which can be reasoned upon automatically. Given the implemented properties and triplet representation, a major improvement to the capabilities of the system would be

[31] https://wordnet.princeton.edu/

[32] https://babelnet.org/

[33] https://engineering.fb.com/2019/01/22/ai-research/laser-multilingual-sentence-embeddings/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Knowledge Graph Embedding (KGE), which would support services such as search, ranking and inference in hierarchical relationships.

KGE would allow for advanced analytics on the data facilitating a host of new operations available to the different pilots in the project, not only in the traditional graph-analytics sense, but also for artificial intelligence operations such as link-prediction and within the scope of DWC, most importantly recommender systems, to overcome the limitations of reinforcement learning. This would allow use to the graph to map a priori knowledge in combination with the embedded data to infer new knowledge.

4. Semantic interoperability for the ALERT system for in situ e.coli and enterococci measurements and machine learning based early warning system for bathing water quality

The semantic interoperability software with the accompanying smart data models as described in chapter 5, has been deployed and tested in the two pilots in Paris. The pilots are both trying to solve different needs and are thus relying on different types of data, formats, and protocols. The early warning software for bathing water quality is an open-source software interface that enables real-time bathing water quality assessment[34].

The middleware in this pilot will coordinate multiple data sources in both a push and pull configuration, interacting with different types of communication protocols and formats. The system relies on real-time data from the SIAAP EDEN system (Environmental Data Exchange), however, for security reason, this data can only be acquired in a push configuration i.e., data can only be pushed from EDEN to whichever system needs to utilize their data. Furthermore, the system also requires real-time freshwater, river flow and other data from different companies and Paris municipality, fetched on demand through different APIs servicing data in CSV, TSV and JSON formats. These data sources must be coordinated and converted to NGSIv2 formatted 'smart' data models supporting semantics, which can enrich the data and thusly provide additional value than the raw data. Therefore, the middleware acts as a coordinator receiving data from EDEN using SFTP protocol, simultaneously fetching other data from various APIs and converting these on the fly, before pushing it into the FIWARE Orion broker, with valid converted smart models. Since the data from the SFTP protocol should support the regular REST calls in FIWARE: GET, POST, PATCH, DELETE verb, the middleware contains a file system hook which detects changes in the file system using a physical file system provider (thus also working in containers), which can detect new files, changes or appends to files and the deletion of said files. These events are thus propagated to the FIWARE instance to ensure that the semantic data is consistent with the data pushed from SIAAP.

[34] https://www.digital-water.city/solution/machine-learning-based-early-warning-system-for-bathing-water-quality/

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

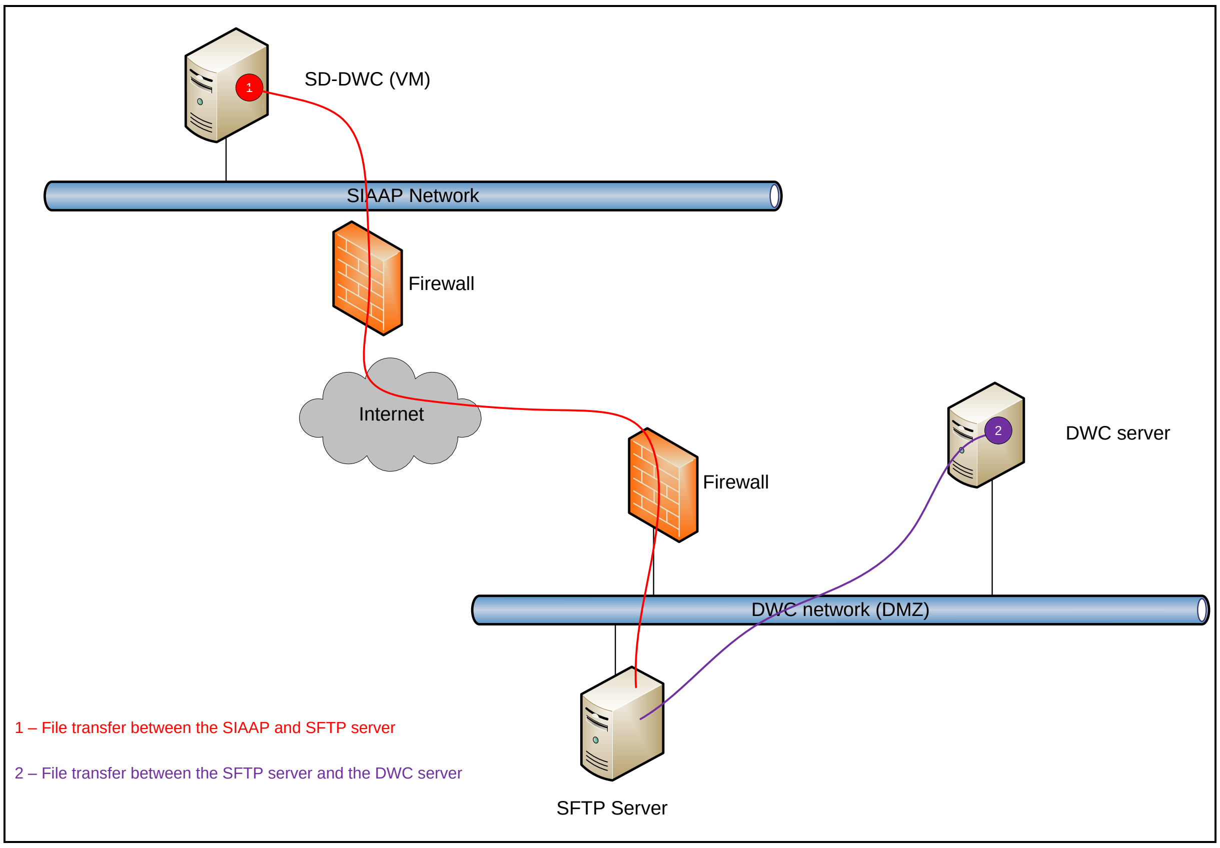

SD-DWC (VM)
1
SIAAP Network
Firewall
Internet
Firewall
DWC server
2
DWC network (DMZ)
SFTP Server
1 – File transfer between the SIAAP and SFTP server
2 – File transfer between the SFTP server and the DWC server

Figure 10: Shows the data flow from EDEN to the middleware

The semantic interoperability middleware contains a map of the physical files and fetched API data, which is managed through a smart model manager, and not until a proper response has been received from the FIWARE instance is the in-memory manage updated. This ensure that the manager has the same state as the FIWARE instance without any polling necessary. This also ensures that the middleware can serialize a map of the order of operations from the remote partner, and that all parties observe the same state.

5. Input to Standardised Data Models

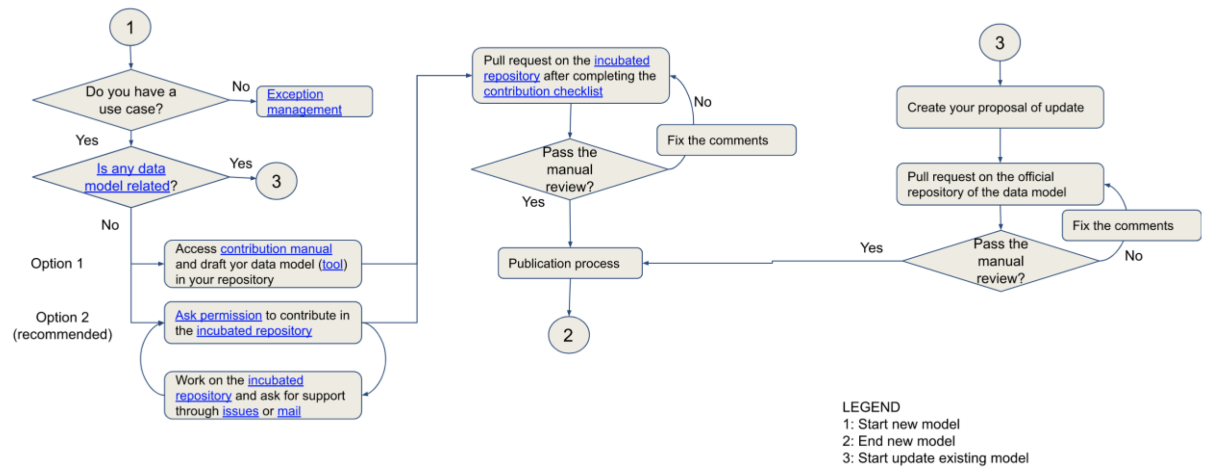
5.1. Procedures for submitting requirements to smartdatamodels.org

Smart Data Models[35] is an initiative that develops standardised data models for several application domains, including water management. These data models are compliant with the NGSIv2 and NGSI-LD formats specified by the standardisation body ETSI, and which are used to represent digital entities in FIWARE Context Brokers. Standardising the data models that are used to describe data within an ecosystem is an important step towards interoperability. The procedures for contributing to

[35] https://smartdatamodels.org/

digital-water.city
digitalwater_eu
digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

smartdatamodels.org are specified in a Contributions Manual[36]. In short, the manual states the following workflow for making contributions:


1
Do you have a use case?
No
Exception management
Yes
Is any data model related?
Yes
3
No
Option 1
Access contribution manual and draft yor data model (tool) in your repository
Option 2 (recommended)
Ask permission to contribute in the incubated repository
Work on the incubated repository and ask for support through issues or mail
Pull request on the incubated repository after completing the contribution checklist
No
Fix the comments
Pass the manual review?
Yes
Publication process
2
3
Create your proposal of update
Pull request on the official repository of the data model
Fix the comments
Pass the manual review?
Yes
No
LEGEND
1: Start new model
2: End new model
3: Start update existing model

According to this workflow, the DWC team followed the recommended path supported by guidelines in meetings with a member of smartdatamodels.org.

Based on the requirements collected in Deliverable D4.4 [1] in DWC we identified a need for one new data model (following Option 2 in the workflow) and extensions to existing data models (point 3 in the workflow).

The new data model, called Water Quality Predicted, is described in Section 5.3.1, while the extensions required from DWC are listed in Section 5.2.

5.2. List of required extensions from DWC

Data Element	Description	Relevant Data Model
UVLampIntensity	Measures the efficiency of UV lamps as milliwatts per square centimetre.	Device Added as Controlled Property Type
UVOrganicLoad	Measures the spectral absorption coefficient of fluids.	Device Added as Controlled Property Type
escherichiaColi	Escherichia coli or E. coli is a type of fecal coliform bacteria that is commonly found in the intestines of animals and humans. E. coli in water is a strong	Water Quality Observed Added as new property

[36] https://docs.google.com/presentation/d/e/2PACX-1vTs-Ng5dIAwkg91oTTUdt8ua7woBXhPnwavZ0FxgR8BsAI_Ek3C5q97Nd94HS8KhP-r_quD4H0fgyt3/pub?start=false&loop=false&delayms=3000#slide=id.p1

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

	indicator of sewage or animal waste contamination.	
enterococci	Enterococci are indicators of the presence of fecal material in water and, therefore, of the possible presence of disease-causing bacteria, viruses, and protozoa.	Water Quality Observed Added as new property
waterLevel	The water level at a certain place in a reservoir, river, storage tank, or similar.	Water Observed Added as new property
waterDischarge	The amount of water discharged from a wastewater treatment plant or stormwater overflows.	Water Observed Added as new property
soilTextureType	Describes the different soil texture types as a set of possible enumeration values: “Clay”, “Clay, Sandy”, “Clay, Silty”, “Loam”, “Loam, Clay”, “Loam, Sandy”, “Loam, Sandy Clay”, “Loam, Silty”, “Loam, Silty Clay”, “Sand”, “Sand, Loamy”, “Silt”. These enumeration values are defined according to the soil texture triangle calculator, defined by the US Department of Agriculture[37].	AgriParcel Added as new property
irrigationSystemType	Describes the type of irrigation system applied as a set of possible enumeration values: “Surface irrigation”, “Localized irrigation”, “Drip irrigation” and “Sprinkler irrigation”, “Center pivot irrigation”, “Lateral move irrigation”, “Sub-irrigation”, and “Manual irrigation”. These enumeration values are defined by the Centres for Disease Control and Prevention[38].	AgriParcel Added as new property

5.3. Smart Data Models

5.3.1. Water Quality Predicted Data Model

[37] https://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/?cid=nrcs142p2_054167

[38] https://www.cdc.gov/healthywater/other/agricultural/types.html

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Since the description of the Water Quality Predicted data model in deliverables D4.4 [1] and D4.5 [3] we have changed the name of the model from Water Quality Prediction to Water Quality Predicted to follow the nomenclature of smartdatamodels.org and added two timestamps `datePredicted` (the timestamp for when the prediction was made) and `expiryDate` (a timestamp for when the prediction is not valid anymore). An example payload of the Water Quality Predicted data model is shown below.

```
 {
    "id": "1024e64a-0283-472c-9b62-dbf77291503e",
    "type": "WaterQualityPredicted",
    "dateCreated": {
     "type": "DateTime",
    "value": "2021-05-20T14:05:00"
    },
    "datePredicted": {
     "type": "DateTime",
    "value": "2021-05-20T14:04:00"
    },
    "expiryDate": {
     "type": "DateTime",
    "value": "2021-05-21T14:05:00"
    },
    "location": {
    "type": "geo:json",
    "value": {
      "type": "Point",
      "coordinates": [48.9159, 2.21228]
    }
  },
    "predictionValues": {
      "value": [
        {
            "percentile": "2.5",
            "prediction": 0.3
        },
        {
            "percentile": "50",
            "prediction": 0.3
        },
        {
            "percentile": "90",
            "prediction": 0.3
        },
        {
            "percentile": "95",
            "prediction": 0.3
        },
        {
            "percentile": "97.5",
            "prediction": 0.3
        }
    ]
},
    "waterQualityPredictionValue": {
        "value": "excellent"
}
}
```

5.3.2. WaterObserved Data Model

The properties waterDischarge and waterLevel have been added to the WaterObserved data model.

```
{
  "id": "some-identifier",
  "type": "WaterObserved",
  "source": "URL to EDEN",
```

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

```
  "dateCreated": "2000-01-01T01:01:01Z",
  "dateObserved": "2021-01-01T01:01:01Z",
  "location": {
      "type": "Point",
      "coordinates": [38.84, 0.105]
  },
  "flow": 8,
  "waterLevel": 2.4,
  "waterDischarge": 3
}
```

5.3.3. WaterQualityObserved Data Model

The properties “enterococci” and “escherichiaColi” have been added to the WaterQualityObserved from Digital-Water.city. Values for these is shown below.

```
{
    "id": "some-identifier",
    "type": "WaterQualityObserved",
    "source": "[URL to Fluidion´s server]",
    "dateCreated": "2000-01-01T01:01:01Z",
    "dateObserved": "2021-01-01T01:01:01Z",
    "location": {
        "type": "Point",
        "coordinates": [38.84084, 0.10513]
    },
    "enterococci": 100,
    "escherichiaColi": 100
}
```

5.3.4. Weather Observed Data Model

Observed weather is relevant in Digital-Water.city and the WeatherObserved data model can be reused as is without the need for any extensions. A sample payload of this data model is shown below.

```
{
  "id": "some-identifier",
  "type": "WeatherObserved",
  "dateObserved": "2016-11-30T07:00:00.00Z",
  "location": {
    "type": "Point",
    "coordinates": [38.84084615227421, 0.10513277488217158]
  },
  "precipitation": 0,
  "source": "EDEN",
  "temperature": 3.3,
  "windDirection": 135,
  "windSpeed": 2
}
```

5.3.5. AgriParcel Data Model

The AgriParcel data model is relevant in the Milan case, and the properties SoilTextureType and irrigationSystemType have been added to this model. A sample payload of this data model is shown below.

```
{
  "id": "urn:ngsi-ld:AgriParcel:72d9fb43-53f8-4ec8-a33c-fa931360259a",
```

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

```
  "type": "AgriParcel",
  "dateCreated": "2021-01-01T01:20:00Z",
  "dateModified": "2021-05-04T12:30:00Z",
  "location": {
    "type": "Polygon",
    "coordinates": [[[45.5791, 9.2921], [45.5695, 9.3021], [45.5616, 9.2764], [45.5620,
9.2620], [45.5791, 9.2552]]]
  },
  "area": 200,
  "description": "Corn",
  "category": "arable",
  "belongsTo": "urn:ngsi-ld:AgriFarm:f67adcbc-4479-22bc-9de1-cb228de7a765",
  "ownedBy": "urn:ngsi-ld:Person:fce9dcbc-4479-11e8-9de1-cb228de7a15c",
  "hasAgriCrop": "urn:ngsi-ld:AgriCrop:36021150-4474-11e8-a721-af07c5fae7c8",
  "cropStatus": "seeded",
  "lastPlantedAt": "2021-04-23T10:18:16Z",
  "hasAgriSoil": "urn:ngsi-ld:AgriSoil:429d1338-4474-11e8-b90a-d3e34ceb73df",
  "hasDevice": [
    "urn:ngsi-ld:Device:4a40aeba-4474-11e8-86bf-03d82e958ce6",
    "urn:ngsi-ld:Device:63217d24-4474-11e8-9da2-c3dd3c36891b",
    "urn:ngsi-ld:Device:68e091dc-4474-11e8-a398-df010c53b416",
    "urn:ngsi-ld:6f44b54e-4474-11e8-8577-d7ff6a8ef551"
  ],
  "soilTextureType": "Clay, Sandy",
  "irrigationSystemType": "Sprinkler irrigation"
}
```

6. Input to SAREF ontologies

To suggest extensions to the SAREF ontology or its extensions (e.g., SAREF4WATER), a change request must be submitted to the committee responsible for maintaining the ontologies. The workflow for submitting change requests is shown in Figure 11. To propose a change request, you must possess a Contributor role in ETSI. A new change request is submitted by creating a new issue that describes the change request in the issue tracker in the ETSI Public Forge[39]. The interaction with SAREF will continue until the project end and beyond to finalize the task based on the feedback of SAREF. Updates will be described in D4.8 at M42.

[39] https://labs.etsi.org/rep/saref/saref-core/-/issues

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

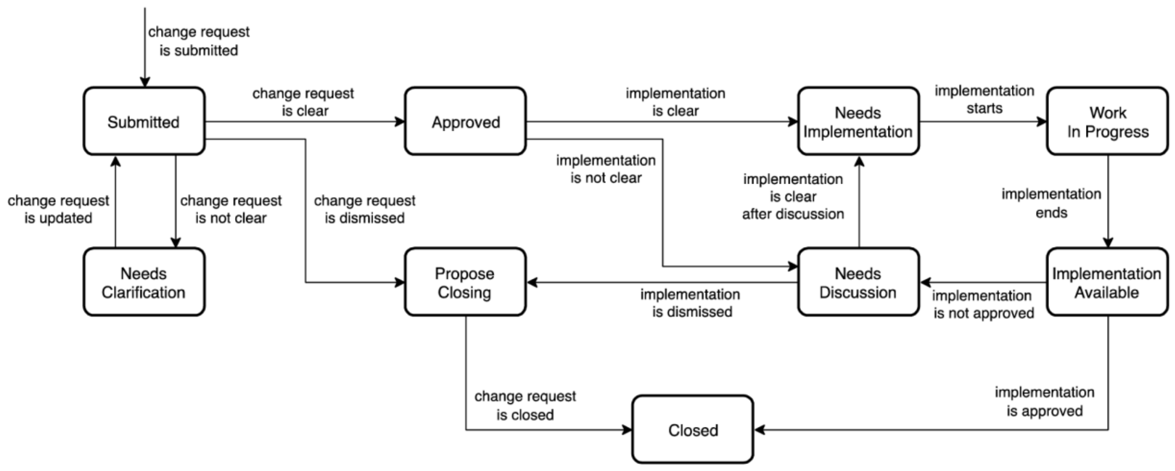

change request is submitted
Submitted
change request is clear
Approved
implementation is clear
Needs Implementation
implementation starts
Work In Progress
change request is updated
change request is not clear
change request is dismissed
implementation is not clear
implementation is clear after discussion
implementation ends
Needs Clarification
Propose Closing
implementation is dismissed
Needs Discussion
implementation is not approved
Implementation Available
change request is closed
Closed
implementation is approved

Figure 11. Workflow for proposing change requests in SAREF[40]

The proposed changes to SAREF4WATR, SAREF Core, and SAREF4AGRI are also described in deliverable D4.5 [3] but are repeated here along with a justification of the change which has been appended to the change request.

At the time of writing this deliverable report, these change requests have been submitted to the technical committee and we are awaiting the next steps of the process (either "Needs Clarification", "Approved" or "Proposed Closing").

Table 1. Summary of change requests submitted to SAREF.

Change request / Issue Title	Description of the change request / issue	Relevant ontology
Water Quality Predicted	*"In the Digital-Water.city project, a use case is using statistical models for predicting the water quality in the river. The predicted measurements should be associated with different percentiles with cardinality 1..* (e.g., a predicted value of 0.3 (in the range 0-1) at percentile 50, another prediction of 0.4 at percentile 97.5) and it should be possible to describe a conclusion from the prediction stating the overall water quality using an enumerated value ("Excellent", "Good", "Sufficient", "Poor"). "*	SAREF4WATR
Sampling Device as a subclass of the existing Device	*"In the project Digital-Water.city there is a need for specifying automated sampling devices and their measurements. A description of such devices is provided here: Automatic Water Samplers. As this might be relevant in other domains besides water*	SAREF Core

[40] http://www.etsi.org/deliver/etsi_ts/103600_103699/103673/01.01.01_60/ts_103673v010101p.pdf

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

	management, the change request is added here."	
IrrigationSystemType	*"There exists a class WateringSystem. However, it could be useful to be able to specify the type irrigation system being used, such as "channel", "drip", "pivot" and "reel". Furthermore, the definition of the WateringSystem class could be made more informative."*	SAREF4AGRI
FarmTypology	*"It should be possible to specify different types of farms. Examples could be CropFarm, LivestockFarm, MixedFarm according to the typology suggested here: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Farm_typology"*	SAREF4AGRI
LandMeasurement	*"There should be a type of property that specifies that a measurement specifies the size of a parcel, a farm or another agricultural entity. For example, when specifying the hectare size of a farm, a measurement could be related to (using the relatesToProperty OP) an instance of a new LandMeasurement class which would then be a subclass of the class Property."*	SAREF4AGRI
Soil Type	*"It could be useful to specify different types of soil, at least at a high level (e.g., whether it is sandy, stony, clay, etc.). Existing classifications of soil types can be found in AgroVoc: https://agrovoc.fao.org/browse/agrovoc/en/page/?uri=http%3A%2F%2Faims.fao.org%2Faos%2Fagrovoc%2Fc_7204" and/or according to the soil texture triangle calculator at USDA: https://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/?cid=nrcs142p2_054167*	SAREF4AGRI
Crop Type	*"It should be possible to specify the type of crops in the ontology. A source of inspiration can be: https://agrovoc.fao.org/browse/agrovoc/en/page/c_1972"*	SAREF4AGRI

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

References

[1] A. Vennesland, P. H. Haro, and B. J. Hanssen, ‘D4.4 Semantic Interoperability Design Requirements’, Digital Water City, 2020.

[2] H. Schwarzmüller, A. Vennesland, P. Halland Haro, and G. Bour, ‘D4.1 Interoperable and secure flow of information: Cyber-physical sphere and interoperability aspects in the utilities regarding the DWC solutions’, Digital Water City, Jun. 2020.

[3] A. Vennesland, ‘D4.5: DWC Water Value Chains Ontology’, 2021. [Online]. Available: https://zenodo.org/record/6497063#.Y8T7TuzMJb8

[4] B. J. Hanssen, P. H. Haro, and A. Vennesland, ‘D4.6: Semantic Interoperable Middleware - Interim Version’, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4320597

[5] A. R. Hevner, ‘A three cycle view of design science research’, *Scandinavian journal of information systems*, vol. 19, no. 2, p. 4, 2007.

[6] R. B. Mishra and S. Kumar, ‘Semantic web reasoners and languages’, *Artificial Intelligence Review*, vol. 35, no. 4, pp. 339–368, 2011.

[7] F. Ekaputra, M. Sabou, E. Serral Asensio, E. Kiesling, and S. Biffl, ‘Ontology-based data integration in multi-disciplinary engineering environments: A review’, *Open Journal of Information Systems*, vol. 4, no. 1, pp. 1–26, 2017.

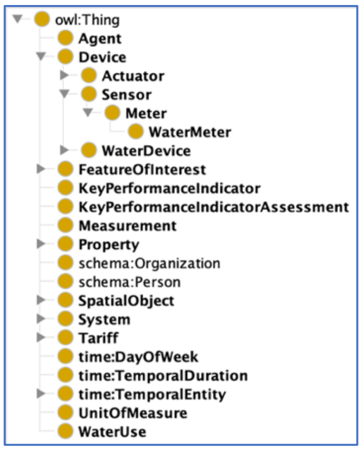
[8] F. Giunchiglia, M. Yatskevich, and P. Shvaiko, ‘Semantic matching: Algorithms and implementation’, in *Journal on data semantics IX*, Springer, 2007, pp. 1–38.

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Annex A: Introduction to semantic models / ontologies

This section provides a minimal and practical description of some key aspects related to ontologies to prepare for the remainder of this report. For a more detailed explanation of ontologies and their application, the reader is referred to e.g., the book Handbook on Ontologies[41] and the W3C Recommendation on the OWL 2 Web Ontology Language[42].

An ontology is a formal definition of the concepts, properties and interrelationships of the entities that exist in some domain of discourse. It provides a shared vocabulary that can be used to describe the domain, classifying and categorising the elements contained within it. Typically, an ontology is formalised using the Web Ontology Language (OWL). OWL is a part of the W3C suite of Semantic Web standards[43], which includes among others Resource Description Format (RDF)[44], a framework for representing web data using subject-predicate-object triples, and the Resource Description Format Schema (RDFS)[45] which provides a data-modelling vocabulary for RDF data. While both OWL and RDFS offer a vocabulary for describing RDF data, OWL allows for greater expressibility than RDFS.



```
owl:Thing
    Agent
    Device
        Actuator
        Sensor
            Meter
                WaterMeter
        WaterDevice
    FeatureOfInterest
    KeyPerformanceIndicator
    KeyPerformanceIndicatorAssessment
    Measurement
    Property
    schema:Organization
    schema:Person
    SpatialObject
    System
    Tariff
    time:DayOfWeek
    time:TemporalDuration
    time:TemporalEntity
    UnitOfMeasure
    WaterUse
```

In an ontology, classes represent sets of individuals (also called instances or objects) with similar characteristics and are organised in a specialisation hierarchy (this hierarchy is also called a *subsumption hierarchy).* This is illustrated in the figure to the right which depicts the specialisation hierarchy of classes in the SAREF4WATR ontology[46]. Here, a `WaterMeter` is a subclass of (specialisation of) `Meter`, `Meter` is a subclass of `Sensor`, and `Sensor` is a subclass of `Device`. This also means that the individuals associated with a particular class are specialisations of those individuals belonging to classes higher in the specialisation hierarchy.

In addition to classes and individuals, ontologies also describe properties, of which there are two fundamental types: object properties and data properties. Object properties define relationships between individuals whereas data properties define literal values associated with individuals. For example, the object property `hasMeasurement` is a relationship that allows for stating various types of measurements of a particular water sample. In the example shown in Figure 9 a sample of water (here, ex:DTSample335632 is an individual of the class `Water`) has a certain concentration of cadmium and e.Coli. The object property `relatesToProperty` allows for defining different types of measurements. The data properties `hasTimestamp` and `hasValue` allows for defining the actual time of measurement and concentrations of cadmium and e.Coli in the water

[41] Staab, Steffen, and Rudi Studer, eds. *Handbook on Ontologies*. Springer Science & Business Media, 2013.

[42] https://www.w3.org/TR/owl2-overview/

[43] https://www.w3.org/standards/semanticweb/

[44] https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[45] https://www.w3.org/TR/rdf-schema/

[46] https://saref.etsi.org/extensions.html#SAREF4WATR

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

sample respectively, while the object property `isMeasuredIn` enables a definition of which unit of measurement is applied.

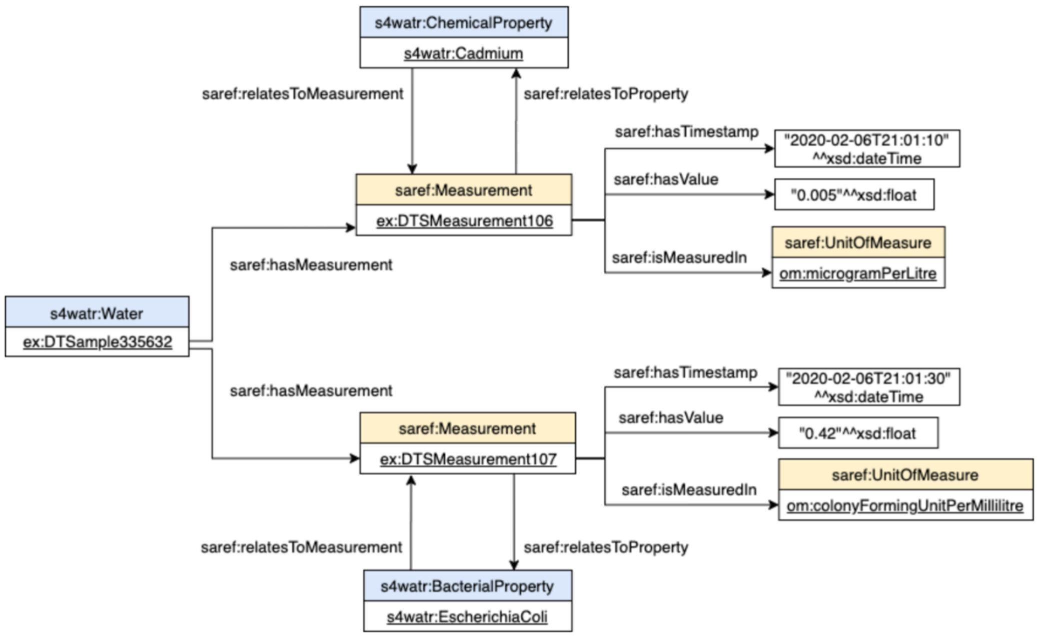


Figure 12. Example from the SAREF4WATR ontology

As revealed by the figure there are two ontologies involved in this example, SAREF4WATER (using the s4watr prefix) and SAREF (using the saref prefix). SAREF is a suite of ontologies[47] where SAREF itself is the core ontology, while there are many extensions (modules) for different application domains. SAREF4WATR is the extension of SAREF for the water management domain. Using SAREF together with one or more of the extension ontologies allows for extending the scope, possibly integrating data from multiple domains into a single knowledge base.

An ontology can be used to uniformly define classes (types), properties (relationships and attributes) and axioms (semantic rules and assertions) of data entities in a knowledge base (aka triple store or knowledge graph). Here, data are described in the triple format (subject-predicate-object) such that according to the example in Figure 12 you would have the following three linked triples stating the measurement of cadmium in a water sample:

Subject	**Predicate**	**Object**
DTSample335632 (type Water)	hasMeasurement	DTSMeasurement106 (type Measurement)
DTSMeasurement106 (type Measurement)	relatesToProperty	Cadmium (type ChemicalProperty)
DTSMeasurement106 (type Measurement)	hasValue	0.005 (datatype float)

[47] An overview of the SAREF suite of ontologies is available at: https://saref.etsi.org/index.html

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Provided that NGSI-LD is used as format for expressing entities in the context broker (e.g., Orion-LD) and associated data storage, quite powerful queries as well as learning techniques can exploit both the explicit (as in the example above) and latent semantics expressed in the ontology.

One example of using latent semantics from knowledge bases is knowledge graph embedding. In the works of Myklebust el al. (2019)[48] knowledge graph embedding techniques are used to model eco-toxological effects of various compounds in the water environment. The idea is that based on the known eco-toxological effects declared in the knowledge graph, the knowledge embedding model will compute/learn the probability of unknown eco-toxological effects. This is also known as link prediction. For example, the knowledge graph states that compound X affects (e.g., has a lethal effect) on species Y. How the compound X affects species Z is not known. But based on the learned vector positions of X, Y and Z, and the quantified effect (relationship) X has on Y, the model also predicts that X affects Z. The quantified effect (relationship) can for example be represented by some computed distance/offset between X and Y. The vector space representation of all entities in the knowledge graph is generated by a so-called knowledge graph embedding model (e.g., based on neural networks). The objective of these models is to learn an optimal vector representation for each entity in the knowledge graph and the intuition is that these vectors capture some latent (unexpressed) semantics from the context of each entity in the knowledge graph. Here, context is represented by for example the structural characteristics of the knowledge graph (e.g., which entities are neighbors to entity E in the graph) or ontological definitions (e.g., entity E is a member of the class Arsenic).

48 Myklebust, Erik B., et al. "Knowledge graph embedding for ecotoxicological effect prediction." *International Semantic Web Conference*. Springer, Cham, 2019.

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.

Digital Water .City

Leading urban water management to its digital future

digital-water.city
digitalwater_eu

digital-water.city has received funding from the European Union's H2020 Research and Innovation Programme under Grant Agreement No. 820954.