D2.1 Specification of system architecture for water consumption and quality monitoring

Author: Fernando López (FF)

Co-Authors: Alberto Abella (FF), Ludovic Bideau (3S), Aitor Corchero (EUT), Stéphane Deveughèle (3S), LLuis Echeverria (EUT), Fernando López (FF), Benoit Orihuela (EGM), Sonia Siauve (OIEau), Chris Sweetapple (Unexe)

June 2020

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant agreement No. 821036.

Disclaimer

Intellectual Property Rights

Project Consortium

Executive Summary

Because FIWARE4Water project aims to develop a socially and business relevant system architecture for heterogeneous entities based in FIWARE technology. It is necessary translate the identified requirements as well as manage the corresponding identified gaps in WP1 in order to create a proper reference architecture. The present report provides a extend analysis of the current legacy system components inside the water sector as well as the transport protocol and data representation system. It will be the base of the description of the corresponding gateways mediation components (a.k.a. IoT Agents in the FIWARE Ecosystem) to translate to a common standard representation of context information based on ETSI NGSI-LD.

The first step will be the creation of the corresponding standard data models aligned between ETSI ISG CIM and ETSI SAREF for representing this context information. To do that, it is mandatory to know how the FIWARE4Water community and beyond can contribute to the definition of new standard data models. As a result of all of these activities, a close collaboration between the ICT4Water cluster projects was established in order to create a common framework of smart data models to be used by all the projects.

Secondly, a deep analysis of the corresponding FIWARE Generic Enablers selected in order to complete the corresponding FIWARE4Water Reference Architecture. This activity mainly was developed to provide the proper status of the implementation of NGSI-LD on the different components by the FIWARE Community and identify the gaps that could be covered by the current project (e.g. IoT Agent – Sigfox, MLOps and/or AutoML, new standard data models for water sector). All of these gaps are started to be covered inside the Task 2.2 and Task 2.3. Moreover, deep collaboration channels were established between FIWARE Community and FIWARE4Water in order to follow the roadmap of implementation of NGSI-LD inside the FIWARE4Water Reference Architecture. The results of this collaboration activity are the publication of the following release of FIWARE Generic Enablers with the NGSI-LD functionality by September 2020 in order to be used by the different FIWARE4Water Use Cases.

Furthermore, the quality monitoring of context information requests the introduction of complex management, probably at the level of the data model or even applying some further CEP over the context information. As a result, it is identified several quality mechanisms to cover the corresponding quality of the provided information and establish the steps for the processes to be defined in the Task 2.2.

FIWARE4Water is focused on providing a reference architecture for serious games. Therefore, it is important to cope with cybersecurity aspects that any production environment need to put in place. Moreover, a set of well-known DevOps processed were analysed just to implement the corresponding FIWARE4Water Reference Architecture. Future work will involve the corresponding implementation of the NGSI-LD components both in the FIWARE community and FIWARE4Water, the utilisation of the corresponding MLOps and AutoML concepts and finally the increase of the corresponding Smart Data Models in the water sector.

Related Deliverables

This document is related to the requirements and conclusions obtained in D1.4 - Gap analysis and final Requirements and in D7.3 - Data Management Plan. The conclusion obtained on this document will be the basis for the activities of extensions of FIWARE Components for Cybersecurity, Big-Data and AI defined in D2.2 as well as the support of water management and quality monitoring use cases defined in D2.3.

The description of procedures, the tutorials as well as the F4W-RA will be the base for WP3 and WP4 and the corresponding demonstrators (D3.1 – D3.5, D4.1-D4.5), and for lessons learned (D3.6, D4.6).

Document Information

Programme	H2020 – SC0511-2018
Project Acronym	Fiware4Water
Project full name	FIWARE for the Next Generation Internet Services for the WATER sector
Deliverable	D2.1: Specification of system architecture for water consumption and quality monitoring
Work Package	WP2: Architecture/Data/Ontology/API/Legacy links/Standards
Task	Task 2.1: Legacy-compatible and cybersecure Fiware4Water Reference Architecture
Lead Beneficiary	10 - FF
Author(s)	Fernando López (FF)
Contributor(s)	Alberto Abella (FF), Ludovic Bideau (3S), Aitor Corchero (EUT), Stéphane Deveughèle (3S), LLuis Echeverria (EUT), Fernando López (FF), Benoit Orihuela (EGM), Sonia Siauve (OIEau), Chris Sweetapple (EXE)
Quality check	Chris Pantazis (NTUA)
Planned Delivery Date	M12 (31 May 2020)
Actual Delivery Date	M13 (24 June 2020)
Dissemination Level	Public (Information available in the Grant Agreement)

Revision history

Version	Date	Author(s)/Contributor(s)	Notes
Draft1	15/05/20	Fernando López (FF), Alberto Abella (FF), Benoit Orihuela (EGM)	First version of the ToC
Draft2	28/05/20	Sonia Siauve (OIEau) and stéphane Deveughèle (3S)	Section about Water Legacy Systems
Draft3	04/06/20	Fernando López (FF), Alberto Abella (FF), Benoit Orihuela (EGM), Lluis Echeverria (EUR), Aitor Corchero (EUT)	Complete contribution to the version
Draft4	07/06/20	Fernando López (FF), Alberto Abella (FF)	Sections I, II, III, IV, V, and VI
Draft5	10/06/20	Fernando López (FF), Aitor Corchero (EUR)	Improve section V.1
Drafr6	11/06/20	Fernando López (FF), Aitor Corchero (EUR), Sonia Siauve (OIEau), Chris Sweetapple (EXE), Alberto Abella (FF)	New section V.12
Draft7	19/06/20	Alberto Abella (FF), Fernando López (FF)	Ref to EPANET and final edition
Draft8	22/06/20	Chris Pantazis (NTUA)	Review
Final	24/06/20	Fernando López (FF)	Final version
Final V2	15/03/21	Fernando López (FF)	A section explaining the EU added-value of task 2.1 has been added, following RP1 review.

Table of content

List of figures

List of tables

List of Acronyms/Glossary

ACL	Access Control List
AI	Artificial Intelligence
API	Application Programming Interface
APN	Access Point Name
AS-i	Actuator-Sensor Interface
AutoML	Automated Machine Learning
BLE5	The 5th Bluetooth Low Energy core specification
BOD	Biological Oxygen Demand
CAN	Controller Area Network
CEP	Complex Event Processing
CI/CD	Continuous Integration/Continuous Deployment
CoaP	Constrained Application Protocol
COD	Chemical Oxygen Demand
CSV	Comma Separated Values
DAST	Dynamic Analysis Security Testing
DevSecOps	Development, Security and Operations
DMP	Data Management Plan
DMR	Dynamic Model Representation
DMZ	Demilitarized Zone
DNP3	Distributed Network Protocol, v3
EDR	Endpoint detection and response
ETSI	European Telecommunications Standards Institute
F4W	Fiware4Water
FAQ	Frequently Asked Question(s)
FIPIO	Factory Instrumentation Protocol to manage remote Input/Output
FQN	Fully Qualified Name
GDPR	General Data Protection Regulation
GE	Generic Enabler
GIS	Geographic Information System
GPV	General Purpose (Water) Valve type

HART	Standard analogue signal 4-20mA, with signal modulation
HDFS	Hadoop distributed file system
IoT	Internet of Things
IP	Internet Protocol
IPSec	Internet Protocol Security
ISG CIM	Industry Specification Group (ISG) for cross-cutting Context Information Management (CIM)
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation – Linked Data
LD	Linked Data
LoRa	Long Range Wide Area Network
LoRaWAN	Low Range Wide Area Network
LP-WAN	Low-power wide-area network, also represented by LPWAN
LSS	Large-Scale Systems
LWM2M	Lightweight Machine to Machine
MFA	Multi-Factor Authentication
ML	Machine Learning
MLOps	DevOps for Machine Learning.
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrowband IoT
NGI	Next Generation Internet
NGSI	Next Generation Services Interface
NGSI-LD	Next Generation Services Interface – Linked Data
OPC UA	Open Platform Communications (OPC) Unified Architecture (UA)
ORDP	Open Research Data Pilot
OWASP	The Open Web Application Security Project
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PRV	Pressure Reducing Valve
PSV	Pressure Sustaining Valve
QCI	Quality of Context Information
RBAC	Role-based Access Control

SaaS	Software as a Service
SAST	Static Analysis Security Testing
SDI	Serial Digital Interface
sFTP	Secure File Transfer Protocol
SHACL	Shapes Constraint Language
SIEM	Security Information and Event Management
SIRA	Strategic Research Agenda
SMTP	Simple Mail Transfer Protocol
SoS	System of Systems
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TSS	Total Suspended Solids
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URN	Uniform Resource Name
VCS	Version control systems
vLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network
wM-Bus	Wireless M-BUS (Meter-Bus)
WPL	Work Packages Leader
WssTP	Water Supply and Sanitation Technology Platform
WWTP	Wastewater Treatment Plant
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
YAML	**YAML** Ain't Markup Language

Introduction

This deliverable defines the reference architecture, based on the FIWARE platform, that will be developed, integrated and deployed in the context of the FIWARE4Water project. In particular, it aims at defining a FIWARE reference architecture in the water domain that can sustain the current user needs, that allows for a full interoperability with the legacy systems, and that can allow for developing new and innovative usages, especially in the field of Digital Twin, AI and Big Data.

The first section introduces the core concepts that will be used throughout the document: the legacy systems that will be integrated with the platform, the data models that will serve as a basis for the context information, the digital twin representation that will bring the dynamic model representation, and finally the vision of a system of systems.

The second section presents in detail the different layers of the architecture, the role of each layer and how they integrate together.

The third section introduces and describes in detail the concept of FIWARE-Ready IoT devices.

The fourth section focuses on the operational aspects of the platform and covers the security aspects from the code to the deployment platform and the integration with the legacy systems, the continuous delivery processes, the envisioned deployment platform and the operational tools needed to operate the platform.

The fifth section details the processes, rules and methodologies that will be used to certify the necessary quality of context information.

The sixth section deals with the management of the data models and details precisely the processes related to the design and registration of a new data model.

I. Core concepts

I.1. Legacy systems

Today, the water sector is facing several challenges, such as climate change, population growth, the need for increasing resilience, and ever-rising customer expectations. At the same time, there is a need to maintain an affordable service and public trust. All of this can be addressed by new digital technologies and tools, and the management of large volumes of data provided by the increasing instrumentation of facilities (infrastructures and networks). These technologies are enabling water utilities to extract information and subsequent knowledge from their legacy systems to enhance decision-making and promote water conservation. These aspects will derive in the development of newer digital services for the water sector. To address the current societal challenges, water managers are more conscious in using big data, AI, decision support systems and automatic steering to drive an efficient management and planning of the water resources. These functionalities require new ways of data integration and knowledge extraction from legacy systems and other IoT systems installed in the water infrastructures. According with the ICT4WATER Action Plan[1] and the Strategic Research Agenda (SIRA) of the WssTP[2],

[1] https://www.ict4water.eu/index.php/tag/action-plan/
[2] https://watereurope.eu/wp-content/uploads/2019/07/Water-Europe-SIRA.pdf

newer smart water technologies and digital assets will bring the water sector to a better inclusion of the society into the water value chain and thus, it will derive into an efficient use of the resources. Therefore, new reference architecture, data integration, curation and knowledge extraction techniques are one of the steps to obtain newer insights from the information stored in the legacy systems.

A legacy system is by definition a method, technology, and computer system or application program already in place. A water legacy system is hence the system developed and used to collect information from different data sources, to transmit them to a data acquisition and storage system and, to process them in order to exploit them (visualization, analyse, publishing balance sheets, sharing information, etc.), as illustrated in Figure 1.

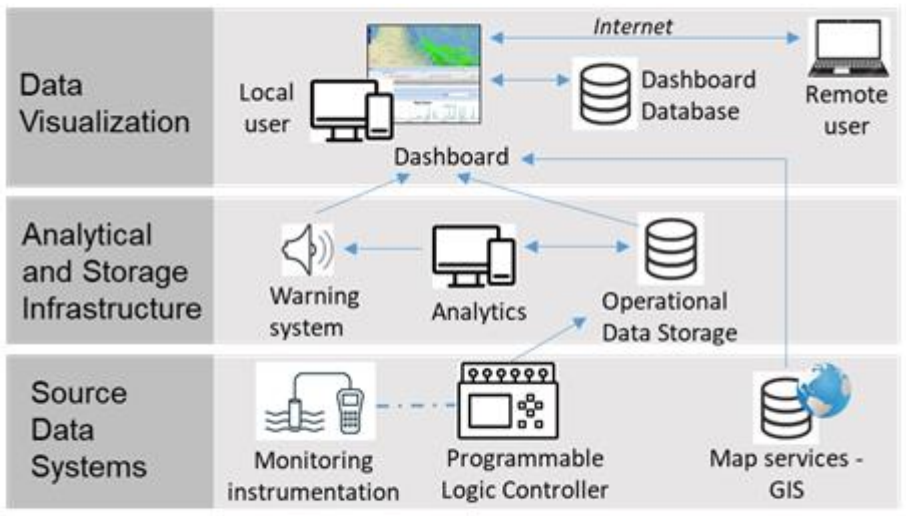

Figure 1: Example of a legacy system for quality monitoring at the exit of wastewater treatment plants

No standards or official guidelines exist to build a water system infrastructure. Each case is unique and the legacy systems currently operating were built to meet a water manager's needs and using available technologies. One challenge is often to later upgrade these legacy systems because technologies and system providers are using different tools with different languages and characteristics. Following is an example of languages, communication systems and tools usually used in the water sector.

The lifespan of a legacy system (software) is generally 10 to 15 years, but changes are regularly made during this period; beyond this period, new services required in operation may require the choice of another legacy system. As for the IT master plans (SDI) for the IS package, they are generally updated every 3 years.

The main obstacle to upgrade a water legacy system is generally the interoperability between all the available technologies and tools available for water digitalization. One of the key advantages of the Fiware4Water platform which system infrastructure is presented in this deliverable is to allow the integration of all the water legacy systems thanks to the development of specific NGSI-LD connectors. The connectors to be developed under FIWARE4Water will serve to ingest the information into a cloud and reference architecture as FIWARE and thus, facilitate the generation of newer digital services and analytics to bring end users and water managers with needed information for the operation and planning of the water critical infrastructure.

Table 1: Example of languages and software commonly used in the water sector

Data acquisition and transmission	Monitoring instrumentation	Various commercial Sensors: acquisition of data and sometimes data processing.
	Wired data transmission to a local Programmable Logic Controller, with a communication protocol	Ethernet (Mobdus TCP/IP, Profinet), serial link (Mobdus, Profibus), HART (standard analogue signal 4-20mA, with signal modulation), AS-i (Actuator-Sensor Interface) and sometimes Ethernet/IP, UniTelway, FIPIO, FipWay, CAN, CANopen, DNP3 …
	Wireless data transmission to a local gateways or concentrator	BLE5, wM-bus …
	Wireless data transmission long range	Cellular networks (2G, 3G, 4G, 5G), LP-WAN (LoRa, Sigfox, NB-IoT) …
Analytical and storage infrastructure	Supervision/SCADA	InTouch, Topkapi, PCVue, Panorama, WinCC …
	Hydraulic models	EPAnet, WaterGEMS, InfoWorks, Mike Urban, …

I.2. Standardization of Smart Data Models

The standardization of the information exchanged between the different stakeholders of the water sector becomes crucial once there is an agreement on the mechanisms of data interchange. These standardization mechanisms are provided by NGSI through an open specification[3]. Data models represent the information objects that interchange between the different agents using the platform. Sharing and adopting an open specification presents massive externalities. Its value actually depends on its adoption. The more users the greater is the value.

The FIWARE foundation together with other partners is driving the Smart data models[4] initiative to standardize data models for being freely used across different sectors. Although, the initiative is paying special attention to the water sector it also comprehends several other domains[5]. Some other data models not directly related with water could have direct impact on the water sector and on the data models to be used (i.e. weather[6], devices[7], etc).

The initiative has adopted a decentralized approach for data modelling allowing relevant actors of the sector to participate in the standardization process. Thus, this initiative by driving a decentralized approach and by using a de facto standardization approach can meet the market speed and requirements.

[3] https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf
[4] https://github.com/smart-data-models
[5] https://github.com/smart-data-models/data-models/tree/master/specs
[6] https://github.com/smart-data-models/dataModel.Weather/tree/master
[7] https://github.com/smart-data-models/dataModel.Device/tree/master

The next paragraph explains why this de facto approach can complement current activities of standardization bodies. Currently, these entities are the most frequent organizations that create standardizations and implicitly they are defining the data models to be used.

However, with the increase of the data produced and interchanged between the agents in the water sector, the standardization process run by these bodies is reaching their limits to provide timely solutions for the demands of the market.

When there were unmet needs, frequently the biggest players of the sector imposed their specifications for the standards. But, as previously mentioned, alternative mechanisms are taking emerging relevance.

One of these mechanisms is ‘de facto’ standardization by ad hoc technical groups usually supported by some organization of the sector. Before further explanation, it is required to define some concepts.

- Standardizing body: Normally a national entity which supports standardization committees (groups) on different topics.
- Committee / working group: Group of experts proposing a standardization on a certain topic.
- Bias of a standard: The resulting standard only adequately reflects the interests of some of the agents involved or affected by the topic.

Comparison between current standardization and ‘de-facto’ standardization

Table 2: Comparison between current standardization performed in standardization bodies and De facto standardization

Concept	Standardization body	De-facto
Composition standardization body	Stable Independent entity.	By topic. Without need of a formal organization.
Prestige	By official review and accreditation mechanism.	For the prestige of its members and its organizations.
Members	Organisations are the members of the standardization body. Their Representatives participate in the working groups.	Experts on the topic, members of relevant organisms in the thematic. At individual level.
Advantages	Control of the composition of the committees and working groups. Review of "polarized" standards Consistency check with other standards Financing guaranteed for the working group.	• Adoption of the standard from the first moment. • Early detection of issues of the by use. • New versions published quickly and easily.
Disadvantages	Very long normalization cycle (years) Publication of the standard behind the needs. Data already coded in alternative ways. New versions require long revision cycles.	• Possible polarization of the standard. • Financing of the working group. • Limited Interest collection of some agents affected by the standard.

The initiative of Smart data models for water capitalizes the concept of ‘de facto’ standardization. It is a highly decentralized organisation where different actors can play a relevant role in some sectors (i.e. water network management) but not to participate in others. The mechanism for standardisation is based on technical groups and the actual use of the data models is required. Thus, pitfalls are detected in very early stages.

Regarding the prestige of the members, the initiative is taking advantage of the critical mass provided by ICT4WATER cluster.

Classification of standardization adoption of data models

Shared data models (standardization) is one of the pillars of data reusability [1]. According to these authors standardization adoption is classified into 4 levels:

1. Own data model standardization
2. Own ad hoc data model standardization published (harmonization)
3. Local standardization
4. Global standardization

The higher the level the bigger the reusability and the easier becomes the adoption. The smart data model’s initiative starts their work on the 3rd level when the data model created is being already adopted and agreed.

Smart Data Models initiative

The Smart Data Models initiative is an initiative to compile, create and curate data models in several business sectors including Water management. The initiative is currently being promoted by FIWARE foundation and TMForum as manager entities, but several other companies and organisations are collaborating in different sectors.

The shared technical assets are published in GitHub (http://smart-data-models.github.com) as a frontend for the technical resources. But it also has a provisional front end for news and mail lists. It is available in the development frontend[8].

Decentralized approach

The Smart Data Models initiative is a collaborative and decentralized initiative about the creation of shared data models. These data models include not only water data models but some other sectors, too. They benefit from cross sector support to expand data models usefulness. For example, weather data models could affect the water forecast for water reservoirs.

The decentralized approach allows the initiative to include relevant organizations in the curation of the different data models. It leaves the leadership of the different sectors to groups which already are relevant in the sector, taking advantage of their knowledge and prestige. The initiative supports the different sectors by providing consistency between the data models, automatic maintenance of the sites of the initiative and a governance model.

[8] http://data-models.fiware.org

Classification of data models

The data models are classified into domains (Smart Water is one of the domains) and every domain compiles several subjects (currently there are four subjects for water: Network management, Consumption, Distribution and Quality). Eventually, new subjects could be created if there are unmet needs. The subjects include related data models and eventually some shared elements common to the data models in the same subject or across different subjects.

Consensus for the creation of data models

The initiative uses consensus as the main decision method, however there are also procedures to solve conflicts and to avoid stopping the evolution of the data models.

Use of existing normalizations

Whenever there is an adopted normalization which has demonstrated to be useful in real case scenario, it is strongly recommended that the data models will follow their recommendations. For example, the data models, collected into the Smart Data Models initiative, include the categories defined in several regulations and standards (e.g. DATEX II in transport, or SAREF4WATR in water domain) to the extent that it does not impact on the needs of the users. This means that when a standardization is against the interest of the users, it can be ignored in the implementation of these data models.

Integration with other platforms

The smart data models' initiative looks for the maximum interoperability with other platforms. In that sense the data models' specification are moving (July 2020) to a YAML specification compatible with the Open API 3.0 with minor modifications. (See Figure 2). This way a user of a data model could on one hand create an entity for NGSI platform, and on the other hand, create the objects to manage with an API 3.0. Not only this but also some exports (i.e. CSV and future SQL) will help developers to profit from the data models for their own developments beyond NGSI but at the same time with payloads that are fully compatible. Therefore, the gateways for connection with these new applications will attend more to the protocols than to the data. Another example is the integration with EPANET software with specific data models adapted to the export for the interaction with this software and that they are already available through a specific repository named water network management[9].

Curation of data models

The consistency of the data models is enhanced by gathering all the different properties defined across different data models and creating a dictionary that can be queried by any contributor. Thus, there is not a need to create a customized property for a data model if there is anything equivalent in any other sector. Furthermore, it provides an additional level of interoperability, not only between data models in a domain but across domains and subjects.

The initiative also provides agile mechanisms of versioning to allow quick evolution of the data models but minimising the need of incompatible versions. In that sense the number of mandatory fields are restricted to a minimum to allow the maximum flexibility on the data model use.

Requirements for contribution

One of the requirements for the contribution of new data models is the existence of actual real case scenarios where the data models are used. The aim of this requirement is to focus on those data models

[9] water network management

that are tested and integrated with other tools (i.e. EPANET software) and therefore desk-design errors are quickly identified and fixed.

Every data model is tested to work on any of the versions of the FIWARE platform and it also includes payloads examples, therefore users have a simple and effortless method to start testing the data models without the bother of creating tests' examples.

Licensing of the data models

The data models are licensed with open license that allow:

- Free use
- Free modifications
- Free sharing of modified versions
- As long as they attribute authorship

In order to release with such licensing a contributor agreement is required to be digitally signed by any contributor. Thus, the Intellectual property owner provides to the initiative the right of releasing the data models with an open license (i.e. Creative commons by, Apache 2.0, etc).

Elements included into a data model

A complete data model includes several elements:

- Specification. A document describing the elements included into the data model. Master version is created in English. The format of the specification (YAML) allows its use with minor treatment into the source for the creation of open API 3.0 objects. See coming features section for further options.
- Schema. A json schema file which validates the key-values payloads of the data model.
- Examples (JSON). Several JSON and JSONLD examples are included. Some of them are in json key values format and others in JSON normalized format.
- Examples (CSV). Although not directly useful for their use in NGSI platform an automatic export of the examples is created to allow interoperability with other systems and platforms.
- Contributors. Globally for every subject there is a record of the contributors to the data models in this subject.
- Current adopters. Examples of implementations of the data models are linked here in order to facilitate the connection with other current users of the data model.

Coming Features

The initiative is under continuous improvement. In the next schema, it is drafted the new automation process for reducing the effort to contribute to authors and users.

By the end of phase 4 a user would be able to create a complete data model (with all the documents described in previous paragraph) in a semi-automatic way.

The main input would be a payload meeting the requirements of a use case. Based on that a draft schema could be automatically generated. Only restrictions and other minor requirements would be necessary to be added to generate the final version. Based on this final schema a template of the specification would be generated with only missing the written description of the properties included into the data model. And finally, based on this schema specification, multiple automatic translations and some additional payload examples could be generated. Both automated actions are aiming at reducing the efforts required to contribute and maximize the usefulness of the data models.

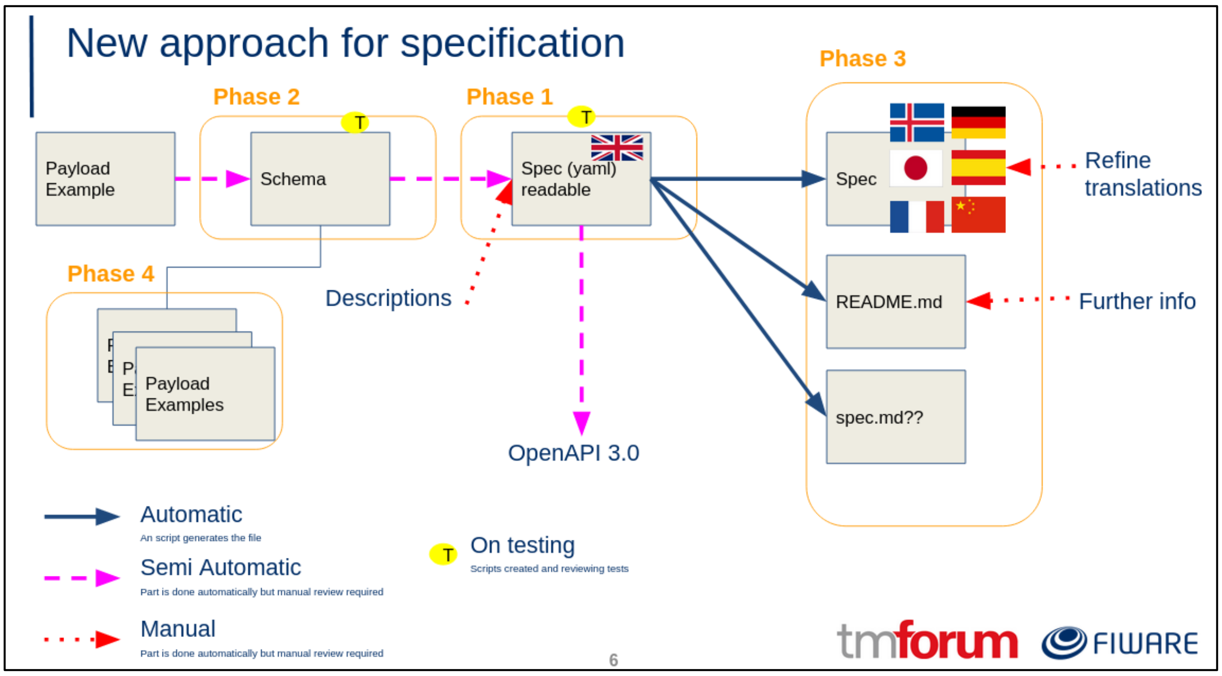

Figure 2: New approach for specification of Smart Data Models

I.3. Digital-Twin representation

We can describe a smart service as an interconnection between context information providers and context information consumers. A provider may be a sensor, a database, an open data repository or even a context consumer that analyse the data in order to provide new context information. They work together in order to create applications to manage, process and notify the information that it is required to offer a service. Of course, each application is associated with the specific environment that it needs to operate and therefore a completely different mix of context from different sources.

This informational representation of something that is supposed to exist in the real world, physically or conceptually is called entity and it is the base of the FIWARE Context Information architecture. Within the FIWARE platform, the context of an entity represents the state of a physical or conceptual object which exists in the real world. For a simple stock management system, we will only need four entity types. The relationship between our entities is defined in the following figure.

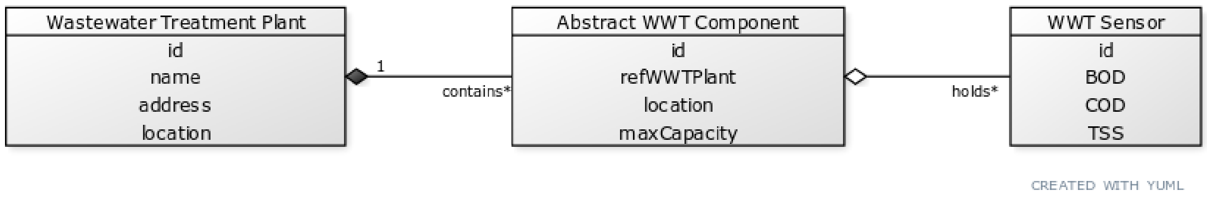

Figure 3: Example of a legacy system for quality monitoring of wastewater treatment plants

- A Wastewater Treatment Plant (WWTP) is a real-world plant to treat wastewater. Plant entities would have attributes such as: + A name of the plant e.g. "KWR Plant 1" + An address e.g. “Australiëhavenweg 15, 1046 BS Amsterdam, Netherlands” + A physical location e.g. 52.3978833 N, 4.7929587,517 E.

- An Abstract WWTP Component is a real-world element in which we proceed with the treatment of wastewater in the different phases. This component has attributes such as: + a location of the component e.g. 52.398869 N, 4.791054 E + a maximum capacity + a reference to the Wastewater Treatment Plant in which this component is included.
- A WWT Sensor is a real-world sensor that measures the quality of the water. This component has the following attributes: + Biological Oxygen Demand (BOD) level + Chemical Oxygen Demand (COD) level + Total Suspended Solids (TSS) level.

Therefore, a WWTP in the real-world is represented in the context data by a WWTP entity, and a real-world WWTP Component found in a WWTP is represented in the context data by a WWTP Component entity which has a refWWTPlant attribute.

The data models allow to create the abstract version and even to validate if a technical representation meets the definition and to reject those representations that include wrong or incomplete values providing a new level of certainty for the managers of the system.

All users of the Internet are familiar with the concept of hypertext links in order to load another content (page) from a known location. While humans are capable of understanding the relationships between different entities, computers are unable to express these deductions without additional information. It is required a well-defined protocol to be able to traverse from one data element to another held in a separate location (e.g. Biological Oxygen Demand or in our example what is the meaning of BOD, where are the measurement units, what are the reference levels, and so on).

Creating a readable links system for computers requires the use of a well-defined data format (JSON-LD) and assignment of unique IDs (URLs or URNs) for both data entities and the relationships between entities so that semantic meaning can be programmatically retrieved from the data itself.

Properly defined linked data can be used to help answer big data questions, and the data relationships can be traversed to answer questions like "What is the COD levels of the WWTP Components of the WWTP X and what is the relationship with the TSS levels on it?"

JSON-LD [2] is an extension of JSON, it is a standard representation format of data to resolve the ambiguity when expressing linked data in JSON format. The data is structured in a format which is parseable by machines. In that way, it is easy to compare all data attributes even if they are coming from different data sources. For example, if two data entities have the name attribute, computers cannot know that both of them can refer to a "Name of a thing" in the same sense (rather than a Username or a Surname or something else). Therefore, URLs and data models are used to remove ambiguity by allowing attributes to have a both short form (such as name) and a fully specified long form (such as an URL like http://schema.org/name) which easily provide the ability for the machines to discover which attributes have a common meaning within a data structure.

Additionally, JSON-LD introduces the concept of **@context** element. The @context element provides additional information to the entities, which allows for the interpretation of the context information by machines (e.g. measurement unit, definition of the concept, mandatory attributes, optional one, etcetera). Finally, the JSON-LD specification enables us to define a unique **@type**. The type allows the association of a well-defined data model to the data itself. In Linked Data, it is common to specify the type of a graph node. This can be obtained based on the properties used within a given object, or the property for which a node is a value. For example, in the schema vocabulary, the diameter property is associated with a Valve. Therefore, one may reason that if a node object contains the property diameter, that the type is a Valve; making this explicit with @type helps to clarify the association.

NGSI-LD [3] is the evolution of the NGSI v2 information model, which has been modified to improve support for linked data (entity relationships), property graphs and semantics (exploiting the capabilities offered by JSON-LD). NGSI-LD has been standardised under the ETSI ISG CIM initiative and the updated specification has been branded as NGSI-LD. The main constructs of NGSI-LD are Entities, Properties and Relationships in the same way that we have in NGSIv2. Nevertheless, NGSI-LD Entities (instances) can be the subject of Properties or Relationships. In terms of the traditional NGSI v2 data model, Properties can be seen as the combination of an attribute and its value. Relationships allow to establish associations between instances using linked data. Moreover, these relationships between entities comprise a more complex data model and more rigid definitions of use which lead to a navigable knowledge graph.

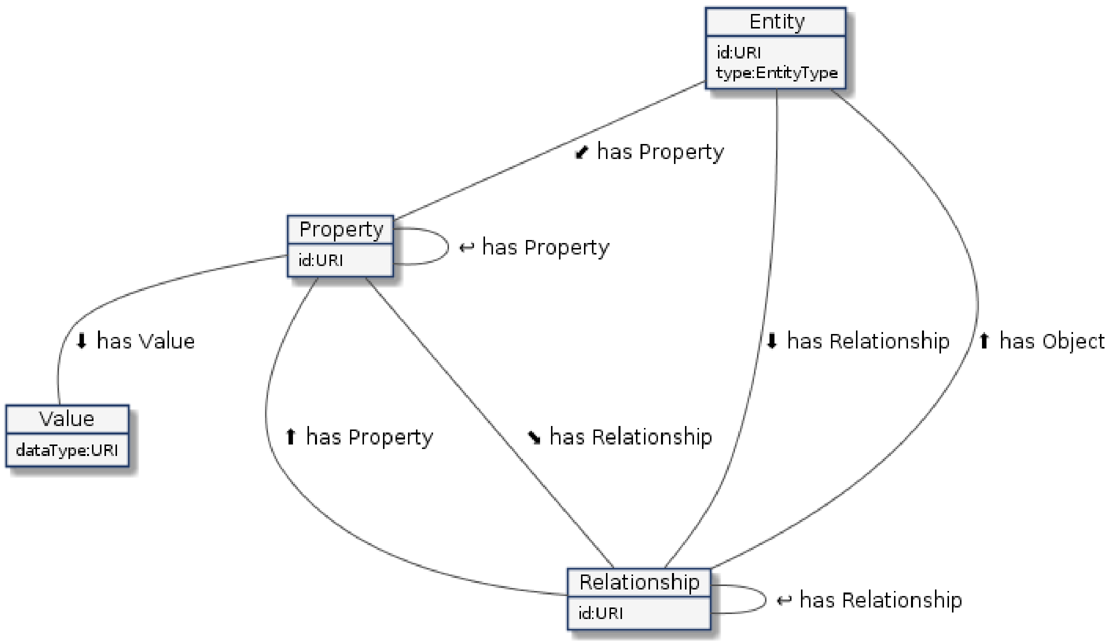

Figure 4: NGSI-LD data model

Once again, entities are the core elements. Every entity must use a unique id which is represented in the format of a URI, often an URN. URIs are also a type, used to define the structure of the data held, which must also be a URI. This URI should correspond to a well-defined data model (e.g. WaterNetworkManagement[10] data model). For example, the URI https://smart-data-models.github.io/dataModel.WaterNetworkManagement/Valve/schema.json is used to define a common data model for a Valve.

If we analyse the Figure 4, entities can have properties and relationships. Ideally the name of each property should also be a well-defined URI. This URI corresponds to a common concept found across the web (e.g. http://schema.org/address is a common URI for the physical address of an item). The property has a value which reflects the state of that property (e.g. name="KWR Plant 1"). Finally, a property may itself have further properties (a.k.a. properties-of-properties). In these cases, properties reflect further information about themselves. Properties and relationships may also have a linked embedded structure (of properties-of-properties or properties-of-relationships or relationships-of-properties or even relationships-of-relationships) which lead to the following:

[10] https://github.com/smart-data-models/dataModel.WaterNetworkManagement

An NGSI-LD Data Entity (e.g. a Valve):

- Has an id which must be unique. For example, urn:ngsi-ld:Valve:valve001,
- Has a type which should be a fully qualified URI of a well-defined data model. Authors can also use type names, as shorthand strings for types, mapped to fully qualified URIs through the JSON-LD @context (e.g. https://schema.lab.fiware.org/ld/context.jsonld#Valve).
- Has a property of the entity, for example, a diameter attribute which holds the valve diameter. This can be expanded into http://schema.org/address, which is known as a fully qualified name (FQN). (e.g. https://github.com/smart-data-models/dataModel.WaterNetworkManagement/blob/master/WaterNetworkManagement-schema.json#/definitions/ngsildProperty).
- Has a relationship of the entity, for example, a valveCurve field, where the relationship valveCurve corresponds to another data entity (e.g. urn:ngsi-ld:Curve:fAM-8ca3-4533-a2eb-12015). It is only required when valveType is equal to GPV.

This example shows us the well-defined knowledge graph and we can expand the relationships indefinitely.

Once we have defined all the static context information, we can move to the representation of the dynamic context information or Dynamic Model Representation (DMR). System models can be represented by different DMR. Once that we have a clear and well-defined governing equation, the system characteristics as well as the response of the system can be modelled. It can be represented as an equation or differential equation and in some cases, in which we have to model a complex system, through the use of a large system of equations.

Furthermore, we need to develop tools that help us model the different DMRs corresponding to each of the Water environments (e.g. a water network) defined in the corresponding data models. These tools take the Context Information provided by the sensors (the static view of the real world), a set of historical data from the sensors and generate the corresponding dynamic view through the execution of the corresponding DMR. In the case of the Water sector, the more widely adopted simulation tool is EPAnet [4], which is used for modelling a water network and performing simulations on it.

FIWARE4Water is defining the corresponding data models to be used in order to easily model the behaviour of the Water systems and integrate the simulation inputs and results in the reference architecture.

Finally, this ability to link the static and dynamic view using NGSI-LD is the core concept of the Digital-Twin. Reliable data becomes the values of properties describing context entities managed at the Context Broker layer, also referred as Digital Twin layer. The Digital Twin is the near-real-time digital image of a physical environment. On the basis of the collected data, the simulation of quality behaviour as well as water distribution systems will be made possible. The goal is to provide a system that is able to keep the data up-to-date with reality, instruct and notify the appropriate changes to the systems (e.g. wastewater treatment plan or water distribution system) in the event of a change in the overall status of the real-world.

I.4. System of systems vision - as mean for data-driven integration of systems

The internet of things (IoT) has a revolutionary potential. A smart web of sensors, actuators, cameras, and other connected smart devices that provide us context information allowing us an unprecedent level of control and automation of **Large-Scale Systems** (LSS).

The subject of LSS involves a multitude of issues of both analysis and design. LSS usually are decomposed into smaller subsystems for controller design or their constituencies are not centrally located together [5]. Large scale systems are common in applications such as chemical process control, power generation and distribution water supply network, among others. LSS can be integrated together to compound more complex systems called **System of Systems** (SoS).

To truly understand the concept of SoS, we could start by looking at how a water distribution plant works. A water distribution plant is an example of this concept. Many systems operate various parts of the plant to get information of the water quality, increase or decrease the pressure of the pipe, and estimate the consume spike of the population. The complete plant is just only work with all its systems work together in a tandem and there is no proper water distribution if these systems work independently of each other.

The emerging SoS concept describes the large-scale integration of many independent and autonomous systems, possibly heterogeneous, but functional, with the main purpose to tackle with a concrete necessity, lower the operational costs and increase the reliability of complex systems. From our water distribution plan, each of the multi-systems deployed are totally independent from each other but each one of them affects the other. The fusion of all these systems often results in different problems mainly related to system interoperability, shared meaning of context information and so on, that are not presented in the design of other single, but complex, systems.

Therefore, SoS is a novel approach for the deployment of integrated, multi-vendor digital solutions that is designed to maximize both maintainability, interoperability, scalability, and sustainability aspects. Attributes like interconnectivity, performance, traceability, and cybersecurity should be considered to generate these intelligent platforms. It is clear that no single company will be able to provide the best solution for all these challenges. Furthermore, the smart water domain is very broad, specialized and diverse. In fact, it is usually needed to count with different solution providers to deal with each water scenario for a concrete use case, which may involve multiple applications with very diverse metering systems working all of them together at the same time.

Moreover, there is a great opportunity to **integrate innovative solutions coming from different stakeholders**. It will be based on the integration of context information generated by different context providers to build a holistic picture of what is going on. As a consequence, Smart Water Management Systems will be able to provide users an integrated and comprehensive view, surrounding context information from different verticals and horizontal solutions. It also covers the integration of third-parties context information (e.g. weather, satellite observations, etc.) through the same common interfaces and data models. In the end, Context Information associated with any water application will be enriched with contributions coming from different vertical solutions (**SoS**), all of them able to share data among each other on a common representation format and enabling a further optimization of processes, saving time, money and resources.

Besides, it is clear that the harmonized data models are the key concept behind the SoS. The availability of shared, well-adopted context information models is a fundamental interoperability mechanism for enabling a global market for IoT-enabled Digital Water Management Applications based on a SoS approach. This allows developing solutions for sector-specific focus while maintaining cross-domain consistency. In particular, **NGSI-LD** is the information meta-model through which concrete, domain-specific data models can be expressed in a coherent way across different domains. Therefore, data models and NGSI-LD are the cornerstone of the FIWARE4Water SoS scheme because they define the harmonised representation formats and semantics that will be used both by water management applications to consume data (through the northbound interfaces) and context providers at the southbound (sensors, existing information systems or open databases) to publish data. Furthermore, data models are one of the key "interoperability points" allowing the participation in a digital single market.

For materializing this ambition, FIWARE4Water is taking into account the concept of "System of Systems" applied to the water sector, where multiple Water Management solutions, based on FIWARE open source components, can be harmonized, combined and managed by a Smart Water Management Systems.

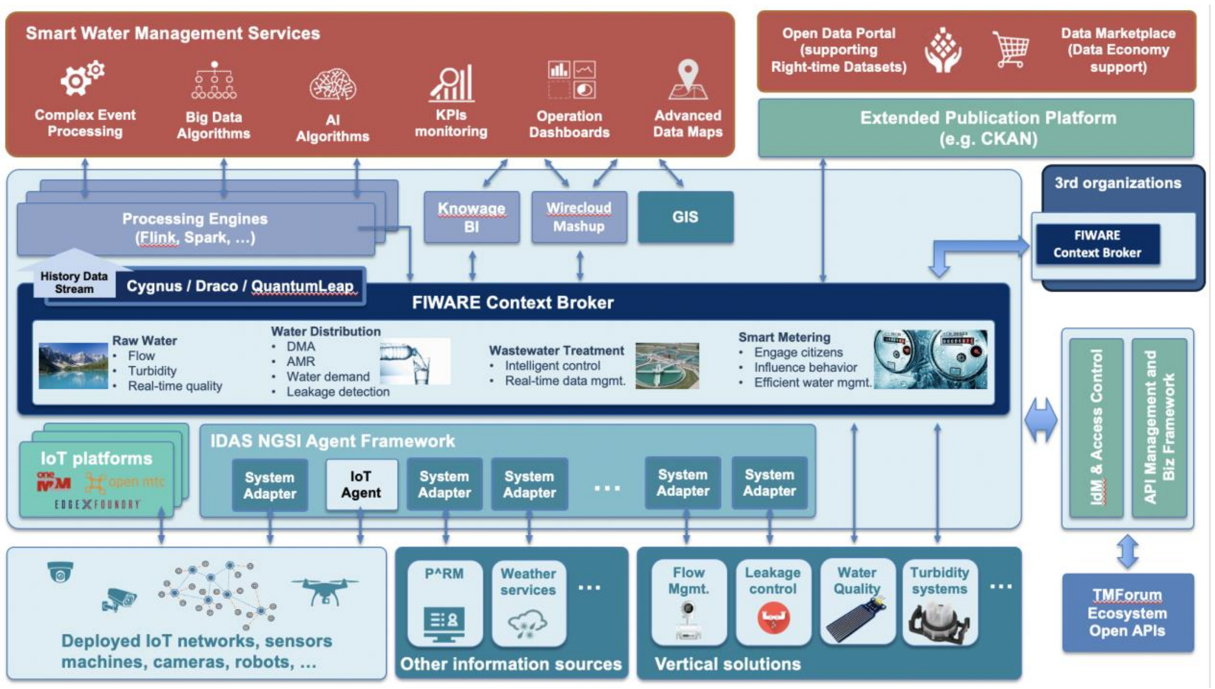

Figure 5: Smart Water Management System (F4W Reference Architecture)

This approach offers us a huge amount of benefits. Firstly, the use of NGSI-LD is the kind of open standard API required for the integration of solutions provided by multiple parties, which avoids the vendor lock-in problem. Secondly, FIWARE already brings a rich suite of open source components (a.k.a. FIWARE Generic Enabler) integrated with the core Context Broker technologies which we will see in more detail in section II – Architecture Layers.

II. Architecture layers

FIWARE (https://www.fiware.org) is a curated framework of open source platform components which can be assembled together with other third-party platform components in order to accelerate the development of Smart Solutions. The main and only mandatory component of any "Powered by FIWARE" platform or solution is a FIWARE Context Broker Generic Enabler, bringing a cornerstone function in any smart solution with the purpose to manage the context information created from one or different context providers and consumed by one or several context consumers.

FIWARE NGSI is the API exported by a FIWARE Context Broker, used for the integration of platform components within a "Powered by FIWARE" platform and by applications to update or consume context information. FIWARE NGSI API specifications have evolved over time and now it is a standard under the umbrella of ETSI ISG CIM group (https://www.etsi.org/committee/cim) which the name ETSI NGSI-LD standard. The FIWARE Community plays an active role in the evolution of ETSI NGSI-LD specifications which were based on NGSIv2 and commits to deliver compatible open source implementations of the specs. The FIWARE Community, with the aid of F4W, plays an active role in the evolution of ETSI NGSI-LD specifications and commits to deliver compatible open source implementations of the specs (e.g. Orion-LD, Scorpio, and Stellio).

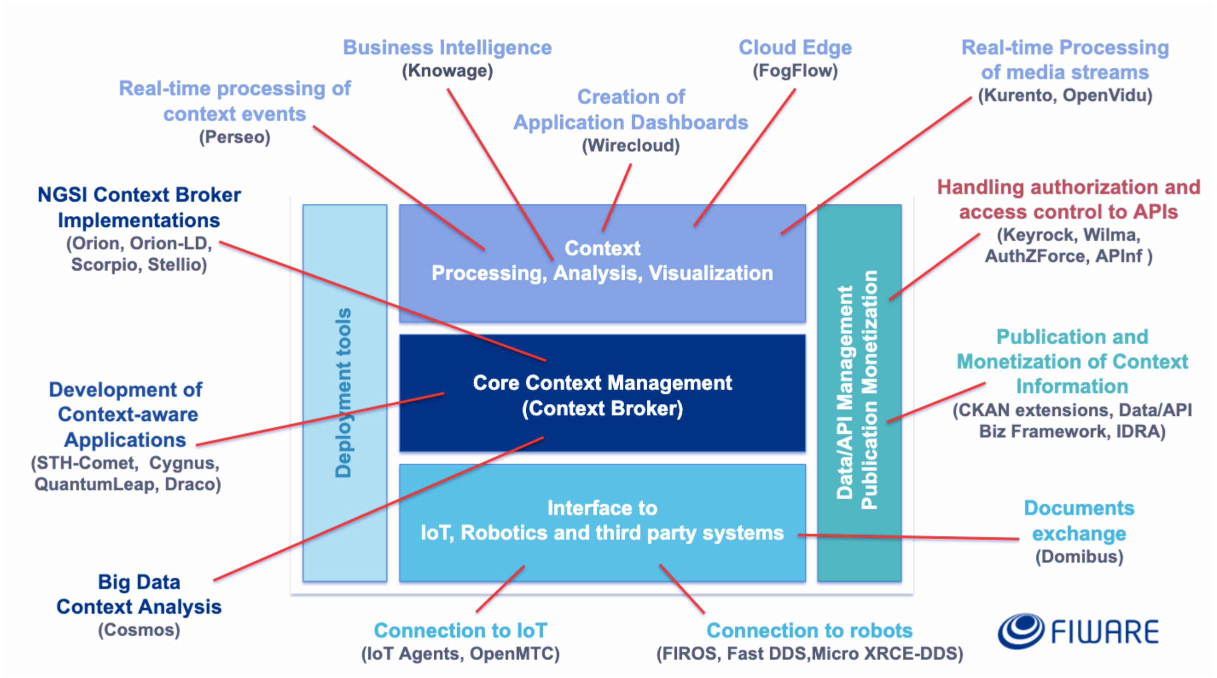
Business Intelligence
(Knowage)
Cloud Edge
(FogFlow)
Real-time Processing of media streams
(Kurento, OpenVidu)
Real-time processing of context events
(Perseo)
Creation of Application Dashboards
(Wirecloud)
NGSI Context Broker Implementations
(Orion, Orion-LD, Scorpio, Stellio)
Handling authorization and access control to APIs
(Keyrock, Wilma, AuthZForce, APInf)
Context
Processing, Analysis, Visualization
Core Context Management
(Context Broker)
Interface to
IoT, Robotics and third party systems
Deployment tools
Data/API Management
Publication Monetization
Development of Context-aware Applications
(STH-Comet, Cygnus, QuantumLeap, Draco)
Publication and Monetization of Context Information
(CKAN extensions, Data/API Biz Framework, IDRA)
Documents exchange
(Domibus)
Big Data Context Analysis
(Cosmos)
Connection to IoT
(IoT Agents, OpenMTC)
Connection to robots
(FIROS, Fast DDS,Micro XRCE-DDS)
FIWARE

Figure 6: FIWARE Catalogue

Building around the FIWARE Context Broker, a rich suite of complementary FIWARE Generic Enablers are available, dealing with the following functionalities:

- Core Context Management manipulates and stores context data so it can be used for further processing.
- Interfacing with the Internet of Things (IoT), Robots and third-party systems, for capturing updates on context information and translating required actuations.
- Processing, analysis, and visualization of context information, implementing the expected smart behaviour of applications and/or assisting end users in making smart decisions.
- Context Data/API management, publication, and monetization, bringing support to usage control and the opportunity to publish and monetize part of managed context data.

FIWARE is not about taking it all or nothing. We are not forced to use these complementary FIWARE Generic Enablers. We may integrate third platform components to design the hybrid platform of our choice. In fact, F4W will be focused on a subset of these components to provide a F4W Reference Architecture based on the ETSI NGSI-LD standard.

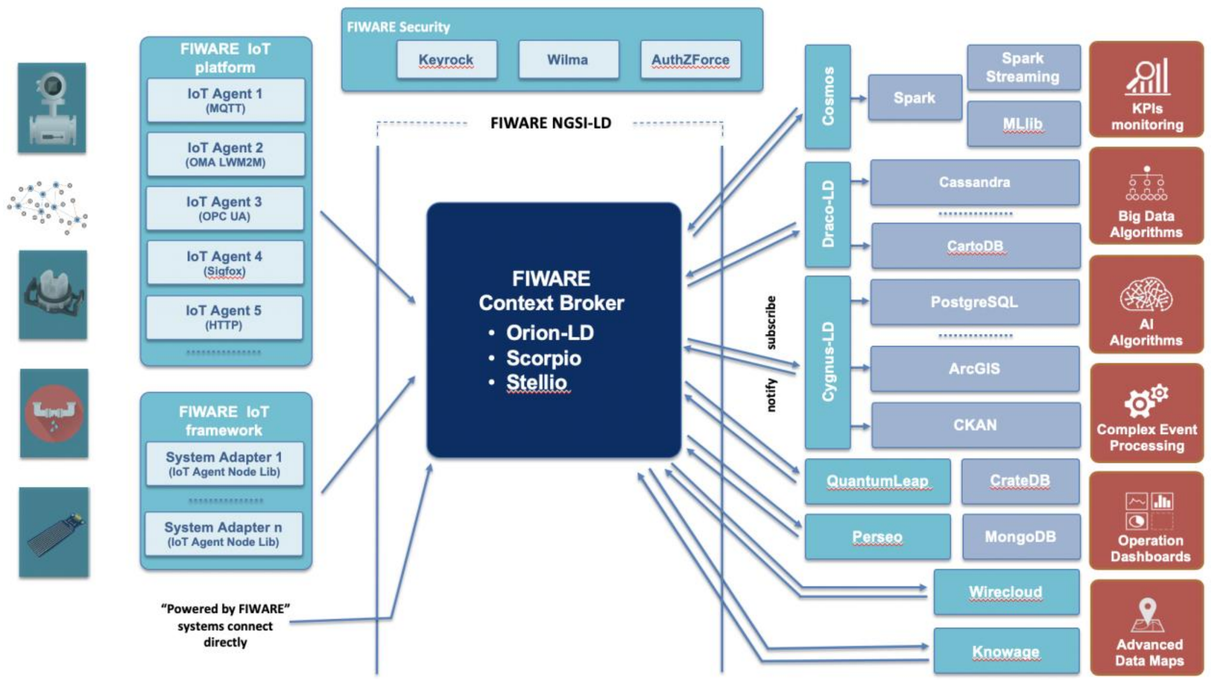

Figure 7: FIWARE4Water reference architecture

This figure provides us an overview of the different FIWARE Generic Enablers that were selected by F4W in order to create the reference architecture based on NGSI-LD.

It is important to mention that as long as any service or solution uses the FIWARE Context Broker technology to manage context information, this platform can be labelled as "Powered by FIWARE" and solutions built on top of it can be labelled as well. The section III describes the concept of FIWARE-Ready IoT Devices and the procedure that has to be followed in order to obtain the "Powered by FIWARE" label.

The section is organized in the following way. Subsection III.1 describes the overall concepts of the generic enablers selected to offer the context information gathering. Subsection III.2 describes in detail the core components of the FIWARE4Water Reference Architecture. Subsection III.3 describes in detail the components used to process, analyse, and visualize the context information. Subsection III.4 describes all the components related to the identity management and access control solutions. Finally, subsections III.5 describes data publication and trading.

II.1. Context Information Gathering: IoT Agents, integration of 3-party systems and legacy systems

A number of Generic Enablers are available, making it easier to interface with the Internet of Things, Robots and Third-party systems for the purpose of gathering valuable context information or trigger actuations in response to context updates. Using sensor data or acting upon these sensors requires an interaction with a heterogeneous environment. These environments are compound by several devices, which are using different protocols, mainly due to the lack of globally adopted standards, accessible through multiple wired and/or wireless technologies.

The main purpose of these components is to provide a gateway to translate those legacy systems, both transport protocol and payload format, to the corresponding NGSI and ETSI NGSI-LD protocol, which is the FIWARE standard API for data exchange model. Of course, we do not need these components if the devices or gateways natively support the NGSI-LD API. In the FIWARE terminology, these components are called IoT Agents.

Additionally, we are able to trigger commands to our actuation devices by updating specific command-related attributes in the associated NGSI entities representation at the Context Broker. This way, all hardware interactions with IoT devices can be handled by the Context Broker, using a homogeneous NGSI-LD interface.

FIWARE Catalogue offers a wide range of IoT Agents, making it easier to interact with the FIWARE technology using them. They are covering the more widely used IoT protocols in the market (LWM2M over CoaP, JSON or UltraLight over HTTP/MQTT, OPC-UA, Sigfox or LoRaWAN).

For the purpose of the FIWARE4Water, we have identified a subset of the entire components that can be used in the F4W Reference Architecture:

- **IoT Agent for JSON** (https://github.com/telefonicaid/iotagent-json) - a bridge between HTTP/MQTT messaging (with a JSON payload) and NGSI/NGSI-LD.
- **IoT Agent for LWM2M** (https://github.com/telefonicaid/lightweightm2m-iotagent) - a bridge between the Lightweight M2M protocol and NGSI/NGSI-LD.
- **IoT Agent for Ultralight** (https://github.com/telefonicaid/iotagent-ul) - a bridge between HTTP/MQTT messaging (with an UltraLight2.0 payload) and NGSI/NGSI-LD.
- **IoT Agent for LoRaWAN** (https://github.com/Atos-Research-and-Innovation/IoTagent-LoRaWAN) - a bridge between the LoRaWAN protocol and NGSI/NGSI-LD.
- **IoT Agent for OPC-UA** (https://github.com/Engineering-Research-and-Development/iotagent-opcua) - a bridge between the OPC Unified Architecture protocol and NGSI/NGSI-LD.
- **IoT Agent for Sigfox** (https://github.com/telefonicaid/sigfox-iotagent) - a bridge between the Sigfox protocol and NGSI/NGSI-LD.
- **IoT Agent Library** (https://github.com/telefonicaid/iotagent-node-lib) - library for developing our own IoT Agent, almost all the IoT Agents are using this library to develop their concrete bridge between legacy systems and NGSI/NGSI-LD.

FIWARE Community offers different tutorials in order to understand the use of the IoT Agents (see https://fiware-tutorials.readthedocs.io/en/latest/iot-sensors/index.html) as well as the development of any needed new IoT Agent based (see https://iotagent-node-lib.readthedocs.io/en/latest) on the corresponding IoT Agent Library. Additionally, FIWARE Community developed a tutorial in order to understand the concept behind linked data and how it is managed by FIWARE components through NGSI-LD (see https://fiware-tutorials.readthedocs.io/en/latest/linked-data/index.html).

Finally, Further information about these components, and other not mentioned here to support NGSI-LD FIWARE4Water Reference Architecture implementation, can be found on dedicated pages provided by the FIWARE Community for IoT Agents (https://github.com/FIWARE/catalogue/blob/master/iot-agents/README.md), Robotics (https://github.com/FIWARE/catalogue/blob/master/robotics/README.md) and Third-Party Systems (https://github.com/FIWARE/catalogue/blob/master/third-party/README.md).

II.2. Context Information Management: Context Broker, Context connectors

The context information management is the key and first topic that every solution has to put on top of the table to be considered a "smart" solution. FIWARE provides the bricks to produce, gather, publish, and consume context information at large scale and exploit it to transform the application into a real smart application. Context information is represented in this environment through values assigned to attributes that characterize the entities used in our application.

Inside this group of components, a Context Broker Generic Enabler is the core and mandatory component of any "Powered by FIWARE" platform or solution. Therefore, it is the core component of our FIWARE4Water Reference Architecture, and all the rest of components are connected to it. It enables to manage context information in a highly decentralized and large-scale manner. For this purpose, the F4W architecture is based on the paradigm of publish/subscribe/notify. It is well adapted to the nature of distributed interaction in large-scale applications. This paradigm allows subscribers to register their interest in an event, or a pattern of events, and be asynchronously notified of events generated by publishers.

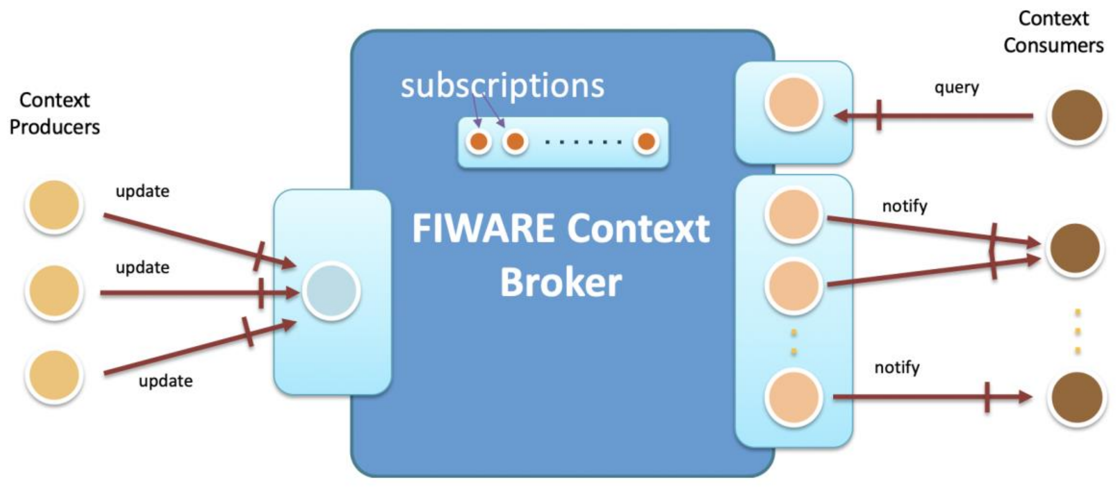

Figure 8: Publish, subscribe, notify paradigm used in F4W RA

Both Orion-LD Context Broker, Scorpio and Stellio Generic Enablers, currently provide the ETSI NGSI-LD API support, which is a simple yet powerful Restful API enabling to perform updates, queries or subscribe to changes on context information based on linked data, how be described in the section I.3.

- The **Orion-LD Context Broker** (https://github.com/FIWARE/context.Orion-LD) Generic Enabler is a NGSI-LD Broker, which supports the NGSI-LD and the NGSIv2 APIs.
- The **Scorpio Broker** (https://github.com/ScorpioBroker/ScorpioBroker) Generic Enabler is an alternative NGSI-LD Broker which can also be used in federated environments based on Apache Kafka technology.
- The **Stellio Context Broker** (https://github.com/stellio-hub/stellio-context-broker) Generic Enabler is another alternative NGSI-LD Broker also based on Apache Kafka technology but including Neo4J graph database for context management as well as timeseries (TimescaleDB) and GIS (PostGIS) database for native historical management.

Accompanying a Context Broker component, as part of Core Context Management, we have a set of components to persist the context information inside databases or make some short-term historical management over the data:

- The **STH-Comet** (https://github.com/telefonicaid/fiware-sth-comet) Generic Enabler allows the persistence of short-term historical context data (typically months) into MongoDB.
- The **Cygnus-LD** (https://github.com/telefonicaid/fiware-cygnus) Generic Enabler allows the persistency of historical context data through the creation of data streams and can be injected into multiple data sinks, including many popular databases such as PostgreSQL, ArcGIS or public Open Data Platform like CKAN[11]. Finally, Cygnus is based on Apache Flume[12].
- The **Draco** (https://github.com/ging/fiware-draco) Generic Enabler is an alternative data persistence mechanism for managing the historical context information. It is based on Apache NiFi[13] and is a dataflow system based on the concepts of flow-based programming. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic and also offers an intuitive graphical interface. Currently Draco supports the persistence of data into MySQL, MongoDB, PostgreSQL, Cassandra, CartoDB and HDFS.
- The **Cosmos** (https://github.com/ging/fiware-cosmos) Generic Enabler allows simple Big Data analysis over context integrated with popular Big Data platforms (Apache Spark[14] and Apache Flink[15]) for stream processing as well as the integration of Machine Learning processing. This is the base component to process the activities associated with the task 2.2.

Last but not least, there is a specific component to persist the context information inside a precise Time-series database (CrateDB) that is an incubated generic enabler within the core context management chapter:

- The **QuantumLeap** (https://github.com/smartsdk/ngsi-timeseries-api) Generic Enabler supports the storage of context data into a time series database (CrateDB and Timescale).

Further information about these components, and others which are not mentioned here and support NGSI-LD FIWARE4Water Reference Architecture implementation, can be found on dedicated pages provided by the FIWARE Community[16].

II.3. Context Information Processing and Visualization

Associated to the Core Context Information components, FIWARE Catalogue provides a set of components developed to process the information and eventually visualize the inferenced knowledge. These components follow the Publish/subscribe paradigm like the other components of the FIWARE Catalogue and basically are context consumers of our smart application. The main purpose, therefore, is making the process, analyse or visualize context information easier in order to provide new context information for the purpose of implementing the "smart behaviour". It is relevant to mention that these context consumers of content information can produce new context information and therefore can be notified as well in our smart application through the FIWARE Context Broker.

[11] It is possible to use NGSIv2 API and in that case the number of sinks is a little more extended. Please take a look on the documentation for more details about the different sinks
[12] Apache Flume: https://flume.apache.org
[13] Apache NiFi: https://nifi.apache.org
[14] Apache Spark: https://spark.apache.org
[15] Apache Flink: https://flink.apache.org
[16] https://github.com/FIWARE/catalogue/blob/master/core/README.md

Inside the FIWARE4Water, we have identified the following components to be used in the F4W Reference Architecture with NGSI-LD support:

- The **WireCloud** (https://github.com/Wirecloud/wirecloud) Generic Enabler brings a powerful web mashup platform making it easier to develop operational dashboards which are highly customizable by end users. Basically, WireCloud allows for the easy creation of web applications and dashboards without programming skills and visualization of context information as well as the control of their environment. This is obtained through the integration of heterogeneous data, application logic, and UI components (widgets) sourced from the Web to allow the creation of coherent and value-adding composite applications.
- The **Knowage** (https://github.com/KnowageLabs/Knowage-Server) Generic Enabler brings a powerful Business Intelligence platform enabling to perform business analytics over traditional sources and big data systems built on context history. High customisable, through the introduction of python code, can provide a wide range of graphic analysis of the data. Knowage is composed of several modules, each one conceived for a specific analytical domain. They can be used individually or combined with one another to ensure full coverage of user's requirements (Big Data, Smart Intelligence, Enterprise Reporting, Location Intelligence, Performance Management, Predictive Analysis and/or Embedded Intelligence).
- The **Perseo** (https://github.com/telefonicaid/perseo-core, https://github.com/telefonicaid/perseo-fe) Generic Enabler introduces Complex Event Processing (CEP) concept using a rules-based system, enabling us to trigger events (send HTTP requests, emails, tweets, SMS messages, etcetera) based on notified information received from the FIWARE Context Broker. This process is executed in real-time, and generates immediate insight, enabling instant response to changing real-world conditions.

Further information about these components, and other not mentioned here, which support NGSI-LD FIWARE4Water Reference Architecture implementation can be found on dedicated pages provided by the FIWARE Community (https://github.com/FIWARE/catalogue/blob/master/processing/README.md).

II.4. Context Data/API management, publication, and monetization

Context Data/API management, publication and monetization brings support to the usage control and the opportunity to publish and monetize part of managed context data. It is divided into two sets of components based on the functionality that they offer, Security Access Components, and API Management, Data Publication and Monetization.

Handling authorization and access control to APIs

Firstly, FIWARE Catalogue offers a set of tools to allow managing the authentication and authorization functionalities in the applications and backend services. It is possible through the use of OAuth2[17] protocol. OAuth 2.0 is the industry-standard protocol for authorization focused on client developer simplicity and providing specific authorization flows (see figure below).

[17] https://oauth.net/2/

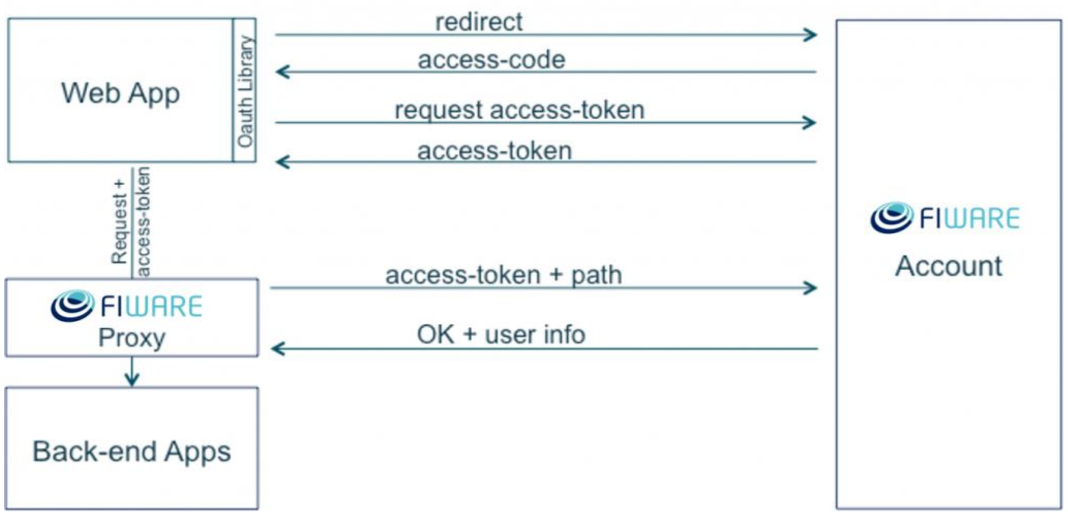

Figure 9: Authentication and basic authorization using FIWARE OAuth2 components

Additionally, it is possible to increase the authorization capability through the introduction XACML (eXtensible Access Control Markup Language) which is an OASIS standard that describes both a policy language and an access control decision request/response language (both written in XML) [6]. This language is used to describe the requirements to access control to context information.

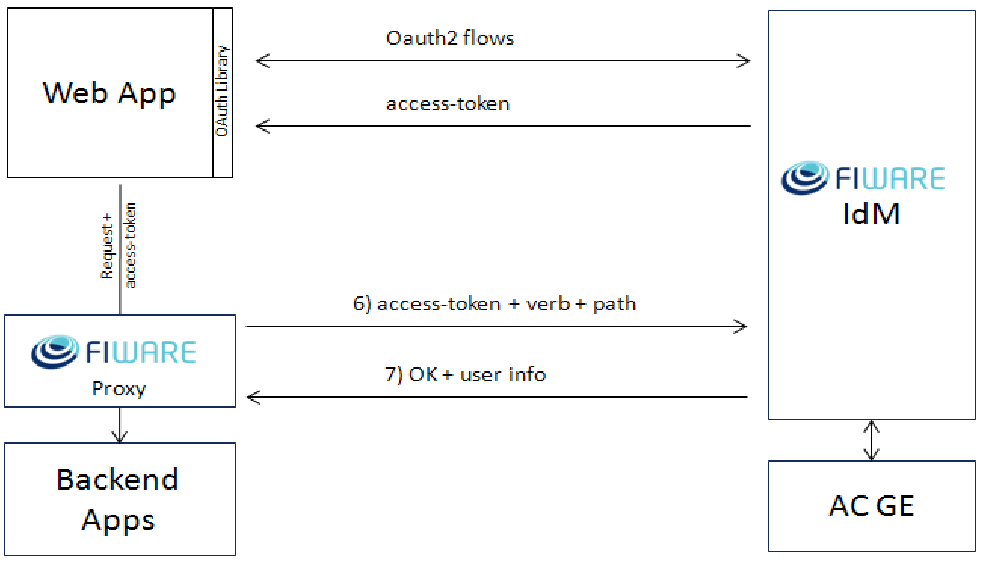

Figure 10: Authentication and authorization using FIWARE OAuth2 and XACML components

The FIWARE Catalogue offers some FIWARE Generic Enablers that implement both the OAuth2 flows and the XACML management and are adopted in the F4W Reference Architecture.

- The **Keyrock** (https://github.com/ging/fiware-idm) Generic Enabler is the Identity Management component that offers secure and private authentication and basic authorization as well identity federation towards applications. Keyrock is a key security component inside the FIWARE System of Systems architecture to offer this security capabilities. It also includes the corresponding tools for administrators to support the handling of user lifecycle functions.

- The **Wilma** (https://github.com/ging/fiware-pep-proxy) Generic Enabler is the FIWARE implementation of the Policy Enforcement Point (PEP) Proxy. This is the component that is located just in front of the backend applications in order to offer authentication and authorization. Therefore, it works together with the Identity Management offered by Keyrock and Authorization Policy Decision Point (PDP) GE offered by AuthZForce.
- The **Access Control** (https://github.com/authzforce/server) Generic Enabler (a.k.a. AuthZForce) provides XACML-standard-compliant authorization services and implements the PDP. According to the OASIS XACML FAQ, "it provides an extremely flexible language for expressing access control that can use virtually any sort of information as the basis for decisions. It is a functional superset of other familiar access control schemes, such as permissions, ACLs, RBAC, etc. It is particularly designed to support large-scale environments where resources are distributed, and policy administration is Federated."

Publication and Monetization of Context Information

Last but not least, F4W Reference Architecture also includes a specific component from the API Management, Data Publication and Monetization list, CKAN Extension. Publishing and consuming open data is a keystone for the development of applications and the creation of an innovation ecosystem. CKAN[18] is one of the most extended Open Data publication platforms and is becoming the de-facto standard for data publication in Europe. Moreover, CKAN is an open source platform which means it can be easily adapted and expanded.

- The **CKAN Extension** (https://github.com/conwetlab/FIWARE-CKAN-Extensions) integrates CKAN solution with the FIWARE platform, enabling the right-time context information served by a FIWARE Context Broker and to be published as a dataset resource, making it easier to be discovered and consumed as Open Data content. Additionally, this extension allows the integration with FIWARE Security in order to enrich the access control and enable explicit acceptance of data terms and conditions, usage accounting, or data monetization. Finally, the integration with WireCloud lets the data providers create and customize rich visualizations for their data, without the need of installing new extensions or restarting the platform.

Further information about these components, and other not mentioned here to support NGSI-LD FIWARE4Water Reference Architecture implementation, can be found on dedicated pages provided by the FIWARE Community:

- Context Data/API management
  https://github.com/FIWARE/catalogue/blob/master/security
- API Management, Data Publication and Monetization
  https://github.com/FIWARE/catalogue/blob/master/data-publication/README.md

[18] https://ckan.org/

III.FIWARE-Ready IoT Devices

III.1. Motivation

Solutions or devices which implement a FIWARE NSGI interface and are able to provide and consume context information but whose architecture is not “powered by FIWARE”, are referred to as “FIWARE-ready”. The ability for a Solution or an IoT device to be "FIWARE-ready" can be supplied via the use of intermediate services such as an IoT Agent which can be used to translate proprietary message formats and transport protocols to NGSI. FIWARE brings open source libraries for development of IoT Agents as well as a portfolio of common IoT Agents which can be used to translate from most popular IoT protocols to NGSI and vice-versa.

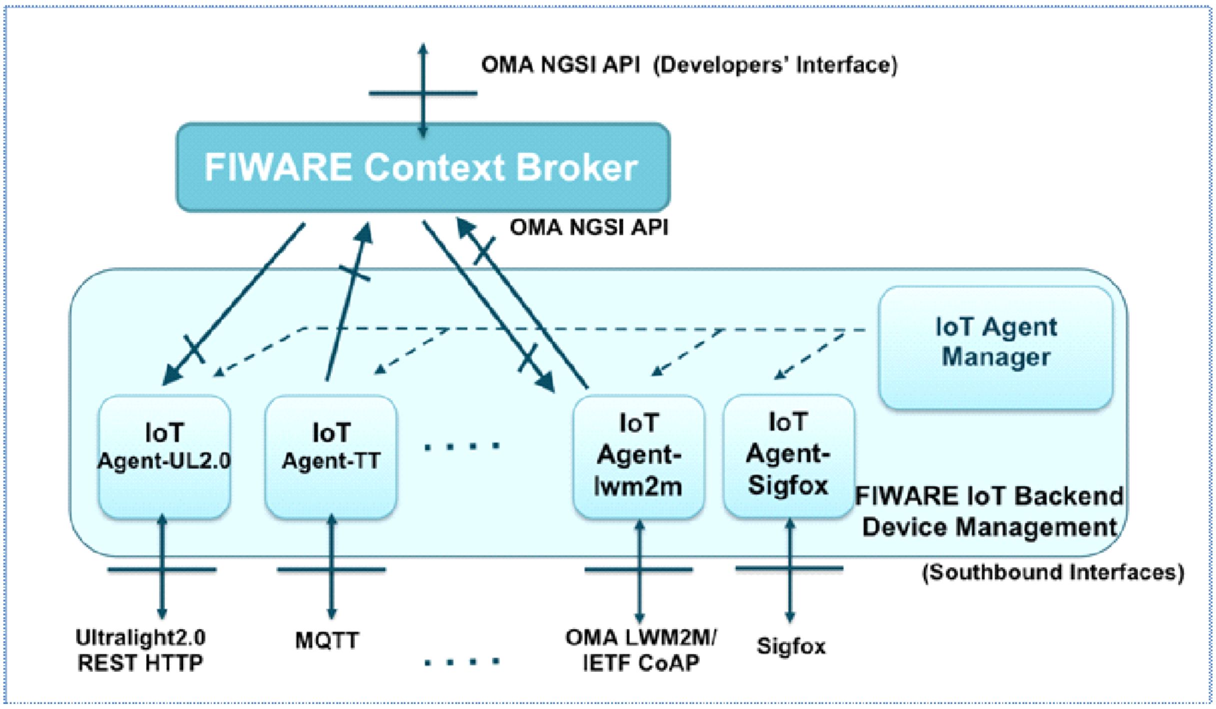

Figure 11: FIWARE-Ready IoT Device testing platform

In summary, the ability for an IoT device to be "FIWARE-Ready" can be supplied either directly - with software on the device or indirectly - via the use of intermediate services such as an IoT Agent (see above) which can be used to translate proprietary message formats and transport protocols to NGSI.

The devices are checked to ensure that readings from sensors can be sent to the context broker and can thereafter be retrieved via the NGSI interface. Actuators are checked to ensure that a change of context made on the Digital Twin held within the context broker results in a real-world action down at the device. This ensures that full operation of the device is possible through NGSI operations only. Further details on how to apply as a FIWARE Ready IoT Device can be found here: https://fiware-marketplace.readthedocs.io/en/latest/device/apply.html.

III.2. Testing Scenarios

The following scenarios are prescribed when validating a device:

Create a Service

Objective: Verify that the implementation is capable of creating a new IoT service.

Applicability: Optional

Pass/Fail Criteria: The new IoT service is successfully created in the context broker.

Register a Device

Objective: Verify that the IoT device implementation has been registered in the context broker.

Pass/Fail Criteria: The context broker sends a status code message indicating that the device has been registered. No error message is received. Thereafter it should be possible to view the digital twin of the registered device within the context broker and access the context data attributes of the registered device.

The logs of the context broker can be checked to ensure that the digital twin has been registered

Get a Device

Objective: Verify that it is possible to retrieve the list of existing devices.

Pass Criteria: The registered devices appear in the list.

For example, the following NGSI-LD call will return all devices registered under a specified {service}.

Table 3: Get a device

```
curl -X GET \
'{context-broker}/ngsi-ld/v1/entities?type={device}' \
  -H 'NGSI-Tenant: {service}'
```

This test ensures that the device state can be accessed using NGSI only.

Send the Measurement

Objective: Verify that the device implementation is able to send measurements

Pass Criteria: The measurements are accessible in the Context Broker. This is testing that a device (and its associated IoT Agent if necessary) is able to communicate using the NGSI protocol with a context broker. Checks can be made by looking at the context broker logs, and the HTTP status code send on each update.

Read the Measurement

Objective: Verify that the device implementation is able to read measurements from the Context Broker.

Pass Criteria: The device implementation is able to retrieve the measurements.

This is similar to the get device test, but it also shows that on-going measurements are received in a NGSI compliant manner as the state of the device (and thus the context) changes.

Table 4: Read the measurement

```
curl -X GET \
'{context-broker}/ngsi-ld/v1/entities?type={device}' \
  -H 'NGSI-Tenant: {service}'
```

Send and Respond to Commands

Objective: For Actuators only, ensure that changes of state made to the NGSI Digital Twin are reflected on the device itself.

Pass Criteria: The device itself is able to respond to commands sent to the context broker

The meaning of "respond" will vary depending on the class of the device itself, but usually it would be expected to show that a physical change has occurred based on the attribute changed (e.g. A lamp switching on when the entity's state attribute is set to "on".

Table 5: Send a respond to commands

```
curl -X PATCH \
'{context-broker}/ngsi-ld/v1/entities/{lamp}/attrs' \
-H 'Content-Type: text/json' \
-d '{
  "state": {
      "type" : "command",
      "value" : "on"
  }
}'
```

IV. Operational aspects

From the user requirements emerged some concerns related to the operational aspects of the platform in general, and to cybersecurity more specifically:

- How to trust Open Source software that is used and integrated into the platform?
- How to deliver an operational, scalable and reactive platform?
- How to ensure the platform stays safe and secure?
- How to monitor the correct behaviour of the platform?

This section is organized as follows: Subsection V.1. describes the quality and security processes to apply during the development, integration, and deployment of components inside the FIWARE platform. Subsection V.2. describes the requirements for a deployment infrastructure that can handle current and future needs of users. Subsection V.3. describes the security measures to apply to a production environment in operation. Subsection V.4. describes the security measures to apply specifically to the communication with the legacy systems used by the pilot sites. Subsection V.5. describes the operation support tools to deploy in order to ensure a correct monitoring of the platform.

IV.1. Secure code, from design to delivery

The first concern relates to the trust and confidence that a user may have in a large platform composed from the development and integration of many Open Source software and libraries.

This is a legitimate concern and the platform has to define and deploy all the necessary processes and tools in order to ensure the maximum level of security in the software delivery chain.

Thus, we are proposing here a set of security practices to be applied from the design of a new piece of software to its delivery in production.

A new term, Continuous Hacking[19], started to emerge recently to design this whole process of ensuring the security chain in software development and delivery. It is associated with the STRIDE acronym: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Escalation. The techniques, processes and tools described below follow and address these security topics.

Secure by design

The first step in this process is to apply the "Secure by design" principles to all the software that is specifically developed in the scope of the FIWARE4Water project. It means that the security is taken into account from the design phase of the application and checked continuously via unit tests focused on security. For instance, if the application receives some user input, it implies to sanitize the data and remove any potential malicious characters.

For this, a minimal and recommended practice is to follow the OWASP Top 10 most critical web applications security risks[20] that directly apply to the phase of code design.

As part of this process, unmaintained and outdated dependencies will also be checked. Indeed, according to a recent survey[21], 82% of codebases have components that are more than four years out of date. This can be a major issue for security concerns, but also for the mid to long term maintenance and sustainability of the platform. Thus, those identified outdated components will be searched for a replacement as soon as possible.

To help in these tasks, the selection and integration of a static analysis security testing (SAST) tool will be realized. A very valuable starting point is the community list of such existing tools that is maintained by the OWASP[22]. Nonetheless, it is expected that the selected tool cover at least the following topics:

[19] https://thenewstack.io/beyond-ci-cd-how-continuous-hacking-of-docker-containers-and-pipeline-driven-security-keeps-ygrene-secure
[20] https://github.com/OWASP/Top10/blob/master/2017/OWASP%20Top%2010-2017%20(en).pdf
[21] https://thenewstack.io/unmaintained-dependencies-and-other-ways-to-measure-ci-cd-security
[22] https://owasp.org/www-community/Source_Code_Analysis_Tools

- Support a rich variety of languages, and at least all the languages used in the components of the platform
- Detect the security vulnerabilities
- Integrate seamlessly in a CI/CD chain

Not directly related to security, but more general to code quality, some nice-to-have features from a static analysis tool are the following:

- Detect potential bugs or "code smells" in the source code
- Perform a global quality check on each release

Dependencies scanning

Nowadays, a typical application or microservice in production has 80% of its source code coming from integrated third-party libraries (which in turn have their own dependencies and so on and so forth).

It is thus very important to integrate a dependency scanning process in order to detect as soon as possible a security vulnerability introduced by one of these third-party libraries. What is more, to be effective, it has to be integrated into the whole software development lifecycle: new source code added, deployment pipeline, external contributions received via a pull request, etcetera.

As of now, some tools have been identified for a careful evaluation (but a larger research will be conducted):

- Dependabot[23], a service provided by GitHub.
- Integrated security alerts in GitHub projects[24], as recently made available by GitHub.
- Snyk[25].

To be valuable, the security scanning of dependencies has to be part of an automated and continuous process, with automatic fixes (or suggestions for fixes at least, via pull request for instance) as much as possible. Thus, it has to be run automatically on a regular basis (for instance, on each pull request, on each commit on the main branches, etc.) and to be followed by immediate actions when this is possible (for instance, a deployment of the platform in production if a critical vulnerability has just been fixed).

DevSecOps

DevSecOps[26] is an extension of the now classical DevOps paradigm. This term is used to emphasize that security must be a core part of the software delivery chain and thus must be deeply integrated into the continuous integration and continuous deployment pipelines.

- For the continuous integration pipeline, it implies at least to cover the following topics:
- Run static analysis security testing
- Run security focused unit tests
- Scan for the security of dependencies
- Secure the Docker containers

[23] https://dependabot.com
[24] https://help.github.com/en/github/managing-security-vulnerabilities/about-security-alerts-for-vulnerable-dependencies
[25] https://snyk.io
[26] https://www.atlassian.com/continuous-delivery/principles/devsecops

The first three points being covered above, the following will specifically address the Docker containers security.

The Docker containers security is a large topic by itself. Docker technology is something relatively new, but very largely widespread. Unfortunately, the security aspect of the containers has not been really addressed from the beginning and there is now a large surface left for attacks. A lot has been done in the past months and there are now mature tools and practices to help in dealing with security in a containerized world. This security field is improving and extending every day, as emphasized for instance by the recent announcement of a partnership between Snyk and Docker[27] to improve the overall security of Docker containers and integrate this concern at the heart of a software delivery chain.

The Docker containers security can be roughly divided into:

- Container creation best practices

  A lot of practices have emerged recently in this field. They range from best practices at the creation time of a container[28],[29],[30] to the need to run Docker containers as a non-root user[31].

  These practices will be thoroughly studied and integrated when wiring up the Docker containers composing the FIWARE4Water platform.

  Complementary to this, tools that help in checking and enforcing these best practices will be used when it is possible to automate the checking (for instance, a tool like Docker Bench Security (https://github.com/docker/docker-bench-security) may be of great value).

  Also, new emerging techniques like Buildpacks (https://buildpacks.io/) from the Cloud Native Computing Foundation will be considered seriously. Indeed, they provide a higher level of abstraction for building apps compared to Dockerfiles and thus bring a new experience into bridging the gap between the source and the Docker packaging of an application and applying best of breed practices in modern container standards. It also ensures that applications meet security and compliance requirements without developer intervention.

- Container security scanning

  As the FIWARE4Water platform will also reuse some existing third-party containers (at least as a base to create new containers for the components of the platform), it is also highly recommended to perform some security scanning on these third-party containers (especially to check they conform to the same security best practices applied at the container creation time), in order not to introduce unexpected security flaws into the platform.

  As of now, some tools have been identified but a larger research will be conducted before a final choice: Clair[32] and MicroScanner[33].

- Image signing

[27] https://snyk.io/blog/snyk-docker-secure-containerized-applications
[28] https://resources.whitesourcesoftware.com/blog-whitesource/top-5-docker-vulnerabilities
[29] https://snyk.io/blog/10-docker-image-security-best-practices
[30] https://thenewstack.io/beyond-ci-cd-how-continuous-hacking-of-docker-containers-and-pipeline-driven-security-keeps-ygrene-secure
[31] https://hub.packtpub.com/docker-19-03-introduces-an-experimental-rootless-docker-mode-that-helps-mitigate-vulnerabilities-by-hardening-the-docker-daemon
[32] https://github.com/quay/clair
[33] https://github.com/aquasecurity/microscanner

In order to bring confidence in the images used, Docker provides tools and practices to apply and check image signing[34], as it is for instance already done in packages distributed on Linux distributions. Such image signing will be applied to each image produced by the platform.

For the continuous deployment pipeline, it implies at least to cover the following topics:

- Dynamic analysis security testing (DAST)

  In the same way that a static security analysis is performed on the source code during the continuous integration phase, a dynamic security analysis is performed during the continuous deployment one.

  Currently, the analysis is performed on the running platform, typically deployed in a dedicated environment, but with a security configuration that has to be the same as the production environment.

  For this specific task, different existing Open Source tools will be evaluated, and a choice will be made for a proven mature solution. Once again, the OWASP site lists some mature solutions and a tool like Zaproxy[35] has already been identified as a serious candidate.

- Penetration testing

  Another very valuable and critical kind of testing is penetration testing. This is a particularly critical point to be addressed for a water management platform that may be subject to cyberattacks, due to the highly sensitive nature of the underlying infrastructure.

  This is a specific field that is well covered and understood. There already exists tools and procedures that will be applied on the platform to be deployed. Security assessment tools, like the aforementioned Zaproxy, can also be used to help and ensure the platform meets the expected security requirements.

- Chaos engineering

  Chaos engineering[36] is quite a new field, not directly related to the platform security, but more to the resilience of the platform.

  It has gained a lot of popularity some years ago when Netflix released the now famous Chaos Monkey[37] project. Aimed at running against a production platform, it tries to “inject” some abnormal behaviour inside the platform (network outage, application failures, …) in the objective to test the application resilience against a bunch of different external or internal factors. Due to the criticality of the software that is going to be deployed on the FIWARE4Water platform, this is an important aspect to be covered.

[34] https://docs.docker.com/engine/security/trust/content_trust
[35] https://www.zaproxy.org
[36] https://principlesofchaos.org/?lang=ENcontent
[37] https://en.wikipedia.org/wiki/Chaos_engineering

IV.2. Deployment infrastructure

The deployment infrastructure must obey to some requirements:

- Be easily deployable by the technical team of the pilot sites on their infrastructure, as they wished to.
- Be responsive as the platform deals with real time data management and processing and users need immediate feedback for decision taking.
- Be highly available as the platform deals with real time data management and processing and users need feedback at any time for decision taking.
- Be scalable as more in more data and usages will come in and the platform must stay responsive and performant over time.

Such concerns are deeply tied to the global architecture of the platform and the components that are integrated. In this respect, it is very important that the platform and its components adhere totally to the Reactive Manifesto[38], which defines the core principles that must be followed by any modern reactive architecture.

Then, it has to be backed by a deployment platform that will bring the ease of deployment, and the tools to allow for high availability and scalability.

Nowadays, Kubernetes[39] is de facto standard for such deployments:

- Deployments can be formalized and automatized, especially via the use of Helm charts[40]
- Integrated support for load balancing
- Integrated support for horizontal scaling
- Automatic restart of containers when a node dies or when a container does not respond to health checks
- Automatic placement of containers based on their requirements

Furthermore, it brings another important feature, by offering to progressively roll out changes to a deployed platform in production, while monitoring application health to ensure all the services are still up for the end users. If something goes wrong, Kubernetes will roll back the changes (whether automatically or manually). This allows for advanced deployment strategies like Blue-Green deployment, Canary deployments, and so on.

Finally, it is expected a tight integration between the CI/CD tool and the deployment platforms (whether production, integration, development, …). Existing tools (like JenkinsX[41]) that permit such a seamless experience, will be considered first (as long as well-known tools in this domain, like TravisCI[42] or Bamboo[43]).

[38] https://www.reactivemanifesto.org
[39] https://kubernetes.io
[40] https://helm.sh
[41] https://jenkins-x.io
[42] https://travis-ci.org
[43] https://www.atlassian.com/software/bamboo

IV.3. Cybersecurity in production

As popularized by the Kubernetes project[44], the security of a cloud native platform relies in the security of the 4C's:

- Security of the code
- Security of the container
- Security of the cluster
- Security of the cloud

The security of the code and of the container are already discussed in detail in the previous sections. The security of the cluster is described in the following paragraphs.

The security of the cloud will have to be evaluated on a case by case basis, as the pilot sites emitted the will to host the FIWARE platform on their own premises or within the infrastructure of their usual cloud provider. Security recommendations will be provided and checked all along the deployment to ensure all the platforms are deployed according to the best practices in cloud security, with support from the technical team of the project.

Security of communications

The security of communications applies to different levels:

- First of all, all HTTP communications have to be done through HTTPS (the use of the Certbot certificate provider[45], which is now well established and largely deployed, will be considered first)
- The communications between the FIWARE platform and the legacy systems will be secured with respect to the security protocols set by each pilot. This is dealt with in the next section.
- The internal communications between the components of the platform should also preferably be secured. This is typically done by using the TLS cryptographic protocol when exchanging data between components, to avoid traffic sniffing
- The communications from and to the sensors, via the IoT Agents. The security of these communications depends on the underlying protocol, so it will be defined and applied on a case by case basis.

Management of secrets

Every microservice has to know some passwords, secrets or tokens to communicate with other systems (be it a database, an external service, an authentication provider, and so on). They of course must not be stored in clear text, not even into a private VCS. A first considered step is to use environment variables defined only on target hosts. A more robust approach is to encrypt secrets and use an external service to manage them (for instance HashiCorp Vault[46] or Spring Vault[47]).

[44] https://kubernetes.io/docs/concepts/security/overview/#the-4c-s-of-cloud-native-security
[45] https://certbot.eff.org
[46] https://www.vaultproject.io
[47] https://spring.io/projects/spring-vault

Slow down attackers

Eventually, an attacker will try to brute force the authentication to the API in order to gain access to the system and expose sensitive or confidential data. One measure to mitigate that risk is to slow down such attacks. This can be done by implementing rate-limiting, whether in the application code or at an API gateway level. It is also more effective if a SIEM tool is deployed inside the platform, for a quicker reaction to such events.

Intrusion detection system

The production also has to be protected from intrusions, that means that an intrusion detection system must be set up (existing tools like Falco[48], Suricata[49] or else Snort[50] will be considered). This eventually can be completed by a SIEM tool.

Data integrity

Finally, the data at rest in databases has to be encrypted, as it can potentially be leaked in case an attacker gains access to the platform. As this is an expensive process, only sensitive or confidential data will be encrypted. The techniques and algorithms depend on each database vendor thus, it will be checked on a per-database basis, and adapted security measures will be applied on each.

IV.4. Security measures in legacy systems integration

The security measures in legacy systems integration must deal with all the communication chain from the captors or sensors to the information systems where all the collected data is stored, analysed, processed and then returned to the users.

For the first step of data acquisition and transmission to the IT systems, the security risk is quite limited because:

- the information is split among the high number of monitoring instruments
- most of the captors or sensors are physically non-easily reachable
- the source data systems are mainly in industrial environments which are the most secured ones: dedicated network with no internet links and very restricted accesses, secured protocols of wired or wireless transmissions (with VPN or private APN if needed).

Even if the water domain is not a privileged target for the cyber-attacks, the security risk increases when the data is reaching the analytical and storage infrastructure and going through to the data visualization.

In those IT environments (specific applications to collect or display data), developers need to fulfil basic technical security rules:

[48] https://falco.org
[49] https://suricata-ids.org
[50] https://www.snort.org

- create separate vLANs for each environment where we can include our servers / VMs depending on their roles. We can create four different vLANs like Backend (the most protected one for the databases for example), Applicative (for the treatments or modelizations softs), DMZ (for the Frontends/APIs exposed on the Intranet or Internet, the files transfers, etc.) and Admin (for administration, support or exploitation components);
- use a strong Active Directory authentication policy to connect to the servers, applicative Frontends or APIs;
- have a security updates policy for the technical components of the servers (Operating Systems, communication protocols, antiviral protection, etc);
- manage a Role-based access control for the reachable applications;
- use secure protocols (e.g. HTTPS, TLS, sFTP) for communications, transfers, URLs, Webservices, … and associated complements if necessary (Reverse Proxy with certificates management, IPSec VPN tunnel, secured SMTP server, internal tokens, etc).

Technical cybersecurity standards are in constant evolution and must be regularly reviewed and improved to fulfil all the new associated requirements like MFA (Multi-Factor Authentication), NextGen Antivirus / Endpoint Detection and Response system (EDR like CrowdStrike for example), proactive vulnerability detection from the source code (using a “tool” like Veracode for example), and so on.

To finish, the GDPR (General Data Protection Regulation) aspects are now a major security requirement to take into account when we implement and manage a complete IT system (personal data recognition and treatment, data retention periods, end user agreement in some cases, etc.).

IV.5. Operation support tools

Operation support tools typically fall into three main categories:

- Monitoring: gather metrics during the runtime operation of the platform, check and ensure they stay in expected behaviour and ranges and notify when something abnormal occurs is about to occur.
- Logging: concentrate in one place all the logs produced by the components of the platform, analyse the messages, inspect past behaviour, and notify when something abnormal occurs.
- Distributed tracing: understand what is happening inside the platform by tracing requests and exchange of information between the components, and thus be able to correlate and follow actions and events.

Monitoring is the most fundamental operation support tool. It allows to monitor the behaviour and the liveliness of the platform at different levels:

- Virtual machines, where it gathers and monitors metrics related to CPU usage, disk space, running processes, I/O, etc.
- Docker containers, where it gathers and monitors individual metrics related to services running in Docker containers: memory, CPU usage, etc.
- Individual services, where the main purpose is to check that services are up and responding in a decent time. It can also be used to monitor internal metrics: HTTP requests received, database requests, etc.

To be efficient, a monitoring tool must be coupled with an alert manager that will be in charge of sending alerts detected by the monitoring tool to a list of recipients on one or more communication channels (SMS, email, …).

From both the monitoring tool and the alert manager, it is expected to be able to define complex alert rules related to one or more of the metrics gathered on the platform, over configurable periods of time. It is also expected that the recipients can be dynamically determined on a per alert basis (there may have a database team that only wants to be notified for database outage, or a specific recipient list for context broker alerts, …). In the same way, not every team or group of people is used to the same communication channels, also pilot use cases already have established alerting practices and the platform has to adapt to them. That is why it is required that the alert manager allows a high level of integration with external systems and communication channels, either directly via email, SMS, Slack, or via a specialized third-party provider like PagerDuty[51] or OpsGenie[52].

Finally, for easier and human friendly access to monitoring information, graphical, real-time, configurable dashboards have to be made available to all users. User access should be integrated with the platform's authentication provider based on the OAuth2 protocol and dashboards and views access should be manageable on a group or role granularity.

In this field, there exists some major Open Source platforms that provide such functionalities (Prometheus[53], Zabbix[54], TICK[55], …). A comparative evaluation will be made before a final choice.

Next comes the logging tools. Modern platforms are typically composed of a set of microservices, that each produce logs. Of course, these logs can be followed and inspected individually but that quickly becomes impractical when there is even a moderate number of microservices. It also quickly makes it difficult to inspect past logs messages to analyse an event.

That is why the platform has to be equipped with a centralized logging tool. It has to allow for an easy integration of the components deployed inside the platform, for instance by supporting common logging formats like Syslog, GELF, Common Event Format or even plain / raw text.

In this field, there exists some major Open Source platforms that provide such functionalities (Graylog[56], ELK[57], …). A comparative evaluation will be made before a final choice.

It is also highly desirable that the selected platform be able to be enhanced for SIEM[58].

Finally, there is the distributed tracing tool. In current modern microservices based architectures, it can be hard to analyse a request, to find what and where has gone wrong, why a request took so long to complete, if we do not have a way to trace its path through all of the micro-services.

This is where a distributed tracing tool comes in, allowing to visualize the path of a request, to see the corresponding logs, the time taken in each microservice, etcetera.

In this field, there exists some major Open Source platforms that provide such functionalities (Jaeger[59], OpenZipkin[60] to name a few). The compliance with the emerging OpenTelemetry[61] specifications from the CNCF is an important factor to consider in the comparative evaluation to be made before a final choice.

[51] https://www.pagerduty.com
[52] https://www.atlassian.com/software/opsgenie
[53] https://prometheus.io
[54] https://www.zabbix.com
[55] https://www.influxdata.com/time-series-platform/telegraf
[56] https://www.graylog.org
[57] https://www.elastic.co
[58] https://en.wikipedia.org/wiki/Security_information_and_event_management
[59] https://www.jaegertracing.io
[60] https://zipkin.io
[61] https://opentelemetry.io

V. Data Quality Management aspects

This section describes the general quality management properties, metrics, and quality extensions of the data models. It also provides an initial mechanism for dealing with Quality of Context Information (QCI) based on NGSI-LD inside the F4W system.

The following sections summarize the current work related to quality of information, i.e. information metadata that describes quality related aspects. The main problem is that the information that we recover from the real world is not always precise. Similarly, data obtained from simulations (e.g. hydraulic and quality data from EPANET), is subject to limitations. Therefore, F4W needs to address some real problems in the measurements obtained:

- How can sensors and actuators depict the quality of the measurements that they are able to provide to the F4W system?
- How can users specify the QCI they require towards the F4W system?
- How can the F4W framework use QCI to select and adapt the resources and their composition to satisfy the request of a resource user?

The first problem requires the definition of a QCI model and a refinement of the publication interfaces. In this section, we provide the requirements and an initial specification for using this model.

For the second problem, our approach is that resource users can specify their QCI requirements on a per basis towards the F4W system through the proper definition inside the data models to be defined in the corresponding Task 2.3. It requires a refinement of the interfaces to manage the QCI. In this deliverable we list the different requirements of the QCI data quality information and the initial specification. It is important to keep aligned the definition of the QCI attributes with the data management plan that is defined in the D7.3 as well as an alignment with the requested QCI in the different demo cases in WP4 and sensors in WP3. The detailed data model definition to cover these requirements will be addressed in a following deliverable D2.3.

QCI is associated with the piece of context information when it is delivered to the resource users following the corresponding defined data model. This means that some requirements should be satisfied in order to allow the quality of the measurements delivered at the same time in which we provide the corresponding context data information. These requirements are described in the sections V.2 to V.11 as a QCI associated with the information model.

Finally, the third problem deals with the mechanism that F4W uses to satisfy the QCI requested by the users or external systems. In particular, it affects the AI as well as ML processes defined in the corresponding task 2.2. The detailed adaptation mechanisms will be addressed in the D2.2 that is going to be delivered later in the project.

This section is organized as follows: Section V.1 describes the alignment with the Data Management Plan (DMP) defined in the WP7. Sections V.2 to V.11 describe the corresponding QCI requirements and the corresponding quality parameters.

V.1. Alignment with DMP

The main objective of this section is to align the terms, datasets, and naming conventions used in the elaborated Data Management Plan (DMP) that has been defined in the Deliverable 7.3 entitled as “Initial Data Management Plan”.

Specifically, the intention is to align the data and dataset generation with the initiative described in the DMP related to the **Open Research Data Pilot (ORDP)**. The main intention with the ORDP is to improve and maximise the access to the generated data under the H2020 project in order to promote furthermore their reusability. To enable this data sharing aspects, the DMP already defines the mechanism to make the data compliant with the **findable, accessible, interoperable, and reusable (FAIR)** principles:

- Findable. The resource can be found in the context through easy mechanisms
- Accessible. Once found that the access can be easily granted with minimal, if any, interaction from the user
- Interoperable. The structure of the data resource meets some shared specification either a standard or a data model coded by the initiative.
- Reusable. The legal right to use the resource is enabled

As this deliverable (description of the architecture) almost deals with datasets and metadata to be exchanged between the FIWARE architectural modules, the next section will describe in detail the type of information and metadata to be applied. Moreover, it also explains the way in which this metadata is stored and shared to be easily consumed by third parties.

Control of the metadata

Considering metadata information to ensure the FAIR principles, the presented architecture will use NGSI-LD connectors in order to describe the properties of the water systems as well as collecting and exposing the data generated under the water infrastructure.

Thus, the metadata information and vocabularies to be used for data exchange corresponds with the NGSI-LD data model (see Section I.3). Complementary, inside the corresponding entity descriptions could be also included some terms coming from the SAREF4Watr ontology.

To share research outcomes, we will use OpenAIRE and the defined guidelines [7] to share the information with the scientific community. According to the documentation, Open Aire platform defines DataCite Metadata Schema v3.1 to share the data. Considering this data schema, the metadata to be considered is the one defined in the following table (Table 6).

Table 6. Open Aire DataCite Metadata fields

Metadata	Field Status
Identifier	M
Creator	M
Title	M
Publisher	M
PublicationYear	M
Subject	R
Contributor	MA/O
Date	M
Language	R
ResourceType	R
AlternateIdentifier	O
RelatedIdentifier	MA
Size	O
Format	O
Version	O
Rights	MA
Description	MA
GeoLocation	O

The field status corresponding with the described metadata is defined in the following table:

Table 7. Field status defined in DataCite Metadata Schema v3.1

Field Status Name	Acronym	Definition
Mandatory	M	The field must always be present in the metadata record. An empty element is not allowed.
Mandatory when Applicable	MA	When the value of the field can be obtained it must be present in the metadata record
Recommended	R	The use of the field is recommended
Optional	O	The property may be used to provide complementary information about the resource

Authorizations of release as open data for ORDP

Deliverable 7.3 points out the need of making data available (Section III.3). The implementation of this feature is a topic under strong debate. On one hand it is important because many examples and use cases require publishing information but on the other hand, there are several alternatives on how to implement it. One of the approaches to this problem is to provide a property that enables all the properties in the payload to be public. But there is another approach in which this would be done individually for every property inside a payload.

Naming conventions:

For those data compiled into datasets (not the data coming from the context broker but for those historical data compiled through other elements of the architecture there is a naming convention (See DMP deliverable 7.3, , section III.2 in page 19) :

F4W_Data_WPx_Tx.x_Name_Vx

Where:

- F4W_Data_WPx_Tx.x_: A prefix for the WP and task in which the dataset has been used/created
- Name: A short and explicit name for the dataset
- Vx : An integer indicating the version of the dataset

Based on the depicted naming convention, the overall data outputs expected for the F4W are the depicted in the Table 8.

Table 8. Data outputs for F4W

Data outputs for Fiware4Water
Citizen awareness and engagement data
Water quality data
Water quantity data
Multiparameter sensor performance data
Wastewater treatment data
Water Demand Forecast data

Simplified sampling procedure

In the case that the payload is related to others, choose a sample of records (5 + number of zeros of the number of records of the payload). For example:

- if there are 1,300 records.

  1,000 (10^3) < 1,300 < 10,000 (10^4) → it will be 5 (fixed term) + 4

- if there are 230,000 records.

  100,000 (10^5) < 230,000 < 1,000,000 (10^6) → it will be 5 (fixed term) + 6

Check that the fields that establish the relationship with other payloads correspond to those found in those payloads. In case of detecting discrepancies, try to find out the cause, document the possible causes and register it for the generation of micro improvement projects.

V.2. Precision, accuracy

Precision of the collected measurements from the demo-sites, previously defined in the corresponding NGSI-LD data model, will be two digits. Indeed, the corresponding connectors that will be developed to deal with the different water infrastructure systems will ensure the measurement transformation into this defined precision by rounding the different values that will be sensed.

Precision indicates the range within which a value provided by a resource can be confirmed true. The definition is adopted from McKeever et al [8] and can be seen basically as a distance in the n-dimensional value space.

On the other side, accuracy indicates the probability of correctness within given precision how it is defined by McKeever et al. Usually, the way in which we provide the accuracy is through a probability distribution function (pdf) and should be compatible between different resources.

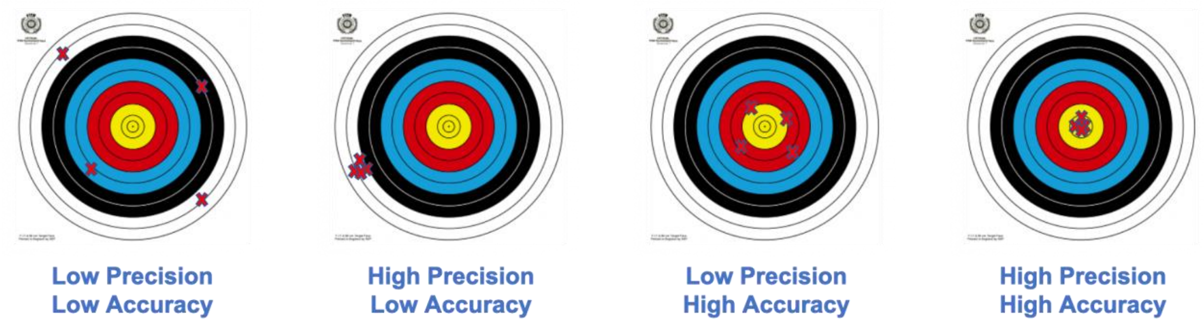

Figure 12: Precision vs. accuracy

V.3. Temporal validation

In terms of temporal representation and validation, the water sector usually represents the dates using ISO 8601[62] date formatting in one of this two versions:

- ISO complete date with time zone designator,
  - Pattern: YYYY-MM-DDThh:mm:ssTZD
  - Example: 2020-05-16T19:20:30+01:00
- ISO complete date with decimal fractions and time zone designator
  - Pattern: YYYY-MM-DDThh:mm:ss.sTZD
  - Example: 2020-05-16T19:20:30.45+01:00

Considering this specific date format to represent the information, we need to differentiate different types of temporal concepts related with the measurements performed by the different digital devices:

- **Measure time** which is the time in which we obtain the corresponding measurement.
- **Temporal scope** is the temporal validity of the measurement [9]. Usually, in the literature, this concept should be called measurement lifetime [10]. The temporal scope can be defined as an exponential decay function of time, in which the temporal validity of the measurement decreases at a rate proportional to its current value depending on the time. Symbolically, it can be represented with the following equation:

  $$M(t) = M_0 e^{-\lambda t}$$

  where M is the measurement, M_0 is the initial value known as the peak amplitude, and λ (lambda) is a positive rate called the exponential decay constant. Therefore, it is only needed to provide the λ value to deal with the temporal scope. By the way, this attribute is optional in the majority of the cases and resources are free to ignore the decay function.

  Additionally, there is another form to represent the equation using the concept of time constant of the exponential, τ. The time constant is the time it takes to decay by 1/e times the initial value.

  $$\frac{M(\tau)}{M(0)} = \frac{1}{e}$$

  It is also known as the mean lifetime or simply lifetime.

  Another important concept is the half-life of the measurement. This is a more intuitive characteristic of exponential decay and represents the decay quantity to fall to one half of its initial value. It is often denoted by the symbol $t_{1/2}$. The half-life can be written in terms of the decay constant, or the mean lifetime, as:

  $$t_{1/_2} = \frac{ln(2)}{\lambda} = \tau * ln(2)$$

[62] https://www.iso.org/iso-8601-date-and-time-format.html

- **Delay time**. Usually, there is a time interval in which the measurement occurs in the real world and the time in which the measurement becomes available in the system [11]. This is also an optional parameter and resources are also free to ignore it.

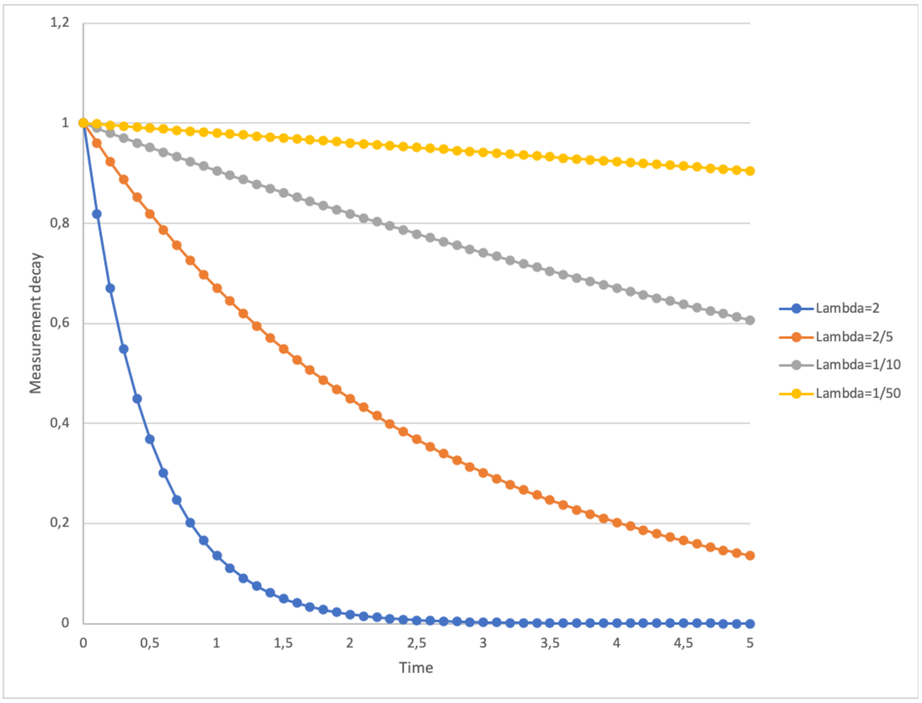

Figure 13: Quantity undergoing exponential decay time

Based on this information exposed above, the architectures and the connectors will always check the consistency of the date formats. For ensuring this consistency, it will be used the following methods:

- Date Format Libraries. Date format libraries in the connectors will be used to check the consistency and ensure the correct date format for the information.
- SHACL language[63]. As most of the information is serialised in JSON-LD, SHACL language could be used to ensure the date is in their corresponding time-window.

V.4. Completeness

Completeness in the data often refers to the degree of all required data available in the corresponding dataset. To ensure completeness in the data, it is required to implement one of the following procedures:

- To check that all consecutive registers exists.
- To check that all required values in the different properties are filled[64].
- To check that the minimum number of elements set for a value is registered and included. This aspect could be easily implemented in the case of standardized data models because such restrictions are included into the schema definition.

[63] https://www.w3.org/TR/shacl/#results-value
[64] https://www.w3.org/TR/dcat-ucr

Completeness has to be checked before using the data for AI or ML processes because missing elements will drive the learning algorithms to bias or just to malfunction. When data is structured into JSON payloads it can be validated with a json schema by using the required clause.

Table 9: Required clause for data model of a Pipe element[65]

```
"required": [
    "id",
    "type",
    "initialStatus",
    "length",
    “diameter”,
    "roughness",
    "minorLoss",
    "startsAt",
    "endsAt"
]
```

V.5. Out of range data (Outliers)

Out of range information commonly can be derived from the failure of the sensors or even the recalibration of the sensors during the time they are being installed in the different points of the water infrastructure. Despite the causes some sensing data is out of range, the result derives in anomalous information that should be fixed at integration time. In these regards, the techniques that could be used are the following ones:

- Semantic data check. Using SHACL rules[66] inside the measurement properties and data schema could serve to determine, for example specific data ranges from temperature sensing or pH sensing values inside specific threshold. As for example, we could apply the following rule to ensure the values of a pH sensor are into a certain range (between 6.5 and 8.5):

Table 10: SHACL rule to assess certain values for a measurement

```
schema:MeasurementShape
   a sh:NodeShape;
   sh:targetSubjectOf schema:hasValue;
   sh:property [
      sh:path schema:hasValue;
      sh:or ( [ sh:datatype xsd:double ] [ sh:datatype xsd:float ] ) ;
      sh:minInclusive 6.5 ;
      sh:maxInclusive 8.5 ;
  ].
```

[65] https://github.com/smart-data-models/dataModel.WaterNetworkManagement/blob/master/Pipe/schema.json
[66] https://www.w3.org/TR/shacl/

Therefore, based on this type of rules applied to JSON-LD models, we can validate the properties and information of the data schema, giving it the necessary consistency.

- Statistically data check. In this regard, outlier's analysis can be used over specific measurements of a sensor type in order to ensure the values are under a certain statistical distribution and thus, ensure a correct distribution of the information along the time series [12]. In the following image, it is depicted how an analysis and correction of outliers could help correct the time series and lately, improve the accuracy of the data analytics processes.

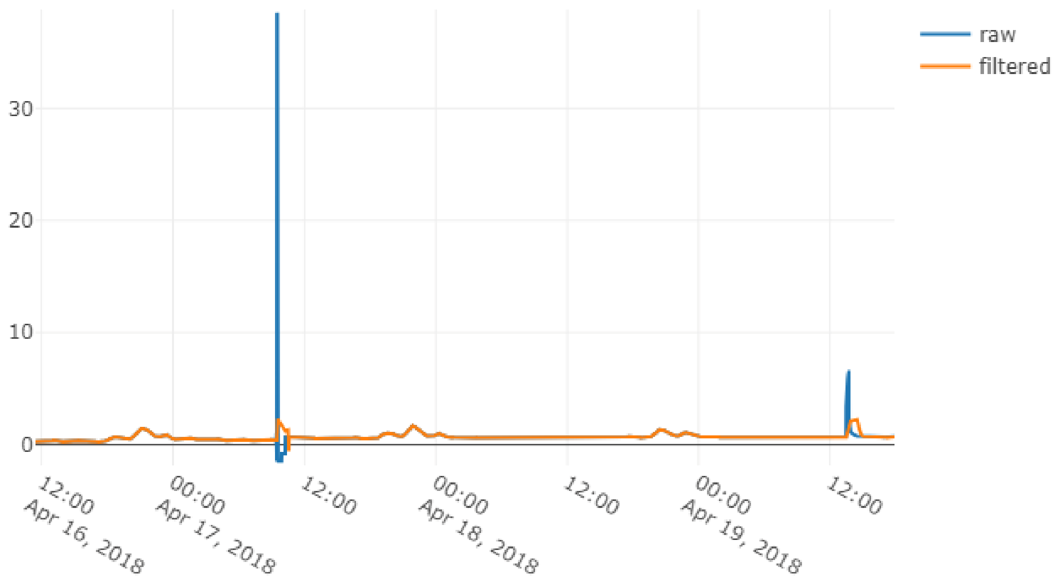

Figure 14: Outlier detection and correction over a time series

One statistical solution to calculate whether or not a point is an outlier, we can use the following equations:

$$x_{Outlier} = \begin{cases} x > Q_3 + 1{,}5 * IQR \\ x < Q_1 - 1{,}5 * IQR \end{cases}$$

Where Q_3 is the Upper Quartile, Q_1 is the Lower Quartile and IQR is the Inter-Quartile Range (Q_3 - Q_1).

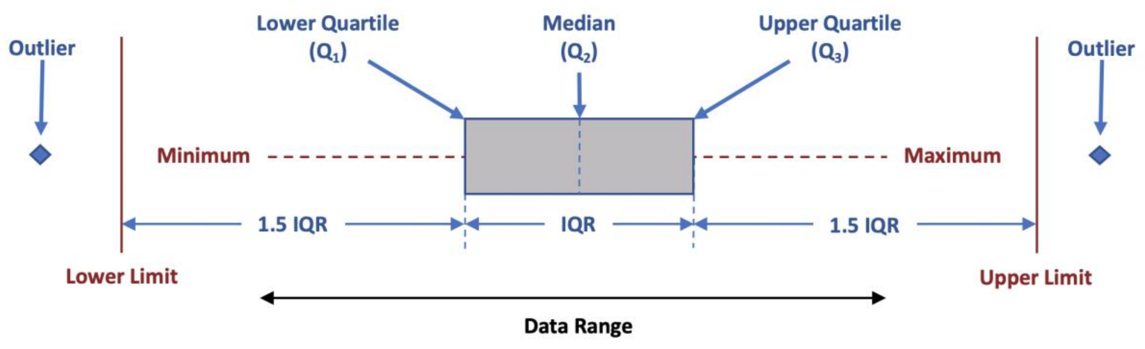

Figure 15: Outliers calculation

V.6. Null / empty values

The null or empty values correspond to some failures in data connection or sensing that impedes gathering a specific measurement from the water infrastructure. Therefore, it will result in a data gap in the corresponding time series. In order to detect and fix the empty values the overall strategy is to search null values in the database and fix them using some of the following techniques:

- Remove the specific data. One of the options is to remove the specific data observation as it generates a noise in the dataset.
- Fix the value. Fix an empty gap could be performed by changing the value using one of the following options:
  - Duplicate the last observation value.
  - Use statistical approximation in order to “forecast” the value following up the same statistical distribution.
  - Perform the mean of the last “N” observations in order to fix the gap.

In case that the null values are not accepted, and that the data are coded into json payloads it can be validated through a clause in the definition of the corresponding json schema. Depending on the type of property it can be validated requiring a minimum number of items in an enumerated value or a minimum length for strings.

Table 11: JSON schema clause to validate properties (null/empty values)

```
Not null for strings
"propertyname": { "type": "string", "minLength": 1 }
...
"required": [ "propertyname" ]

Not null for enumerations
"propertyname": {
    "type": "enum",
    "items": {
        "type”: "string",
        "minitems": 1
    }
}
...
"required": ["propertyname"]
```

V.7. Geolocation correctness

Normally, geolocation is represented in JSON using GeoJSON[67] or using one of their extension to linked data, as GeoJSONLD[68]. Despite the representation used, the resultant file is similar as the exposed in the following listing.

Table 12: GeoJSON representation of the information

```
{
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [125.6, 10.1]
    },
    "properties": {
        "name": "Dinagat Islands"
    }
}
```

Considering this notation to represent geospatial information, geolocation correctness is quite required because of the ease of this kind of error when managing this information. Commonly, the main error data sources on geolocation could come from:

- Wrong geo-codification of text addresses
- Misplacing latitude and longitude when available
- Wrong codification for geographical data

In order to detect geo-location errors, some procedures could be implemented in terms of:

- Detection of location error (out of bounds). Commonly out of bounds geolocation errors fix the latitude and longitude in the 0 (Guinea Gulf). Therefore, checking this location can indicate some errors occur with the geo-referencing of the dataset.
- Using Geospatial queries considering the bounds and context of the demo-sites will also serve to detect and fix some error in the geo-location of the sensors and the corresponding datasets.

Additionally, one of the more important QCI is the spatial quality ones in which we can differentiate between:

- Spatial scope in which our quality context information has a validity only in a specific physical area. It could be expressed as a point, circle, polygon or a multi polygon structure within a given reference system. In example in MongoDB, it is possible to launch a query to determine if a point

[67] https://geojson.org/
[68] https://geojson.org/geojson-ld/

(i.e. coordinates [3,4] is inside an area ([[0,0],[6,0]],[6,5],[0,5],[0,0]]), a rectangle in this case, and it effectively is located into the area.

- Origin location is the physical location, usually related to the sensor location that provides the context information and could be represented like the spatial scope. Similar to the spatial scope, an origin location may be represented as a point, circle or polygon or even a symbolic area specified inside a reference system. Although it is possible to provide a clear outdoor location based on standards method (e.g. using GeoJSON), there is no standard method to provide a location within a reference system, mainly in indoor location (e.g. the location of a sensor inside a tank).

V.8. Duplicates

In the event that the payload does not allow duplicates, we could order the payload by various criteria and verify that there are no duplicates.

In case of detecting discrepancies, we could try to find out the cause, document the possible causes and register it for the generation of micro improvement projects.

NOTE: An easy way to check this case for simple and non-structured payloads that can be traced to a single table is to create a pivot table and group and count for each field.

Table 13: Example for simple detection of duplicates

```
Database 1
id, name, position
1, valve10, [10.3, 4.5]
1, valve10, [10.3, 4.5]
2, valve11, [20.1, 34.2]
3, valve12, [10.3, 4.5]

Aggregated by id
id, count(id)
1, 2
2, 1
3, 1
```

V.9. Inconsistent replicated data

One of the main sources of inconsistency are the replication of data. So, the principles to follow in order to reduce this kind of errors are these:

- When possible in the design phase, replication of data between data repositories it has to be avoided.

- Whenever this is not possible, a valid 'source of truth' has to be set (a master copy of the data), while the rest of data has to be dependent and in case of conflict the master copy rules.
- This last case is not possible, then implement another mechanism to check which copy is the one more accurate (i.e. by including an update date together with the information).

For example, when storing historical data about the sensor these have to be distributed in different databases or repositories for the different analysis to be performed. As previously stated, some wrong values could be gathered due to sensors malfunction, errors on the communication mechanisms, etc. Correction could be extended to some of these replicas of the databases but not to others creating inconsistent data coming from the same sensor in the very same moment.

Table 14: Inconsistent data across replicated databases

```
Database1
id, name, position
1, valve10, [10.3, 4.5]
2, valve11, [20.1, 34.2]
3, valve12, [10.3, 4.5]

Database2
id, name, position
1, valve10, [4.3, 5.5]
2, valve11, [20.1, 34.2]
3, valve12, [10.3, 4.5]

Database3
id, name, position
1, valve10, [4.3, 5.5]
2, valve11, [20.1, 34.2]
3, valve12, [7.3, 1.0]
```

V.10. Wrong codifications

Some of the properties of the payloads could be coded for any reason (an id coming from external databases, because it belongs to codified field according to an external regulation). This type of error comes when there is not an online connection with the sources of coding, or the regulation is updated for some other reasons. Some of the possible solutions to resolve this issue include:

- Masks for input is a good measure to reduce this kind of error in the ingestion process whenever it is manual.
- Redundancy characters (checksum) for some of the critical properties is recommended.
- Background processes checking the validation of coded properties needs to be implemented, especially for those critical properties.
- Wrong codifications are frequently found when different measurement systems are used.

The example below shows a field which is calculated based on the sum of the individual figures of the code field. In the example for registers 1 and 2 is right but for the register 3 is wrong.

Table 15: Example of checksum property in a database

```
Database1
id, code, checksum
1, 01034, 8  (0+1+0+3+4 =  8 Checksum is right)
2, 29120, 14 (2+9+1+2+0 = 14 Checksum is right)
3, 43021, 7  (4+3+0+2+1 = 10 Checksum is wrong)
```

V.11. Out of normalized data

In the event that the schema declares that any of the fields is normalized, carry out a sampling (with the protocol defined above) and check compliance with the normalization. In case of detecting discrepancies, try to find out the cause, document the possible causes and register it for the generation of micro improvement projects.

This kind of error is frequently found when there are possible categories for a data and there is not the control of capitals and therefore the same category can be codified as (Pattern, PATTERN, pattern, etc).

Table 16: Out of normalization values in a database

```
Database1
id, type, name
1, pattern, valve10  (pattern = pattern, right)
2, Pattern, valve11  (Pattern ≠ pattern, wrong)
3, PATTERN, valve12  (PATTERN ≠ pattern, wrong)
4, Pattern, valve13  (Pattern ≠ pattern, wrong)
```

V.12. Simulation options and settings

For data that is the result of a simulation, options and settings used in the simulation will affect the quality of the results. Water distribution network hydraulic and water quality data derived from EPANET simulations is affected by the selected simulation time steps convergence, tolerance and checking options [13]. Specifically, these include:

1. Hydraulic time step (how often the hydraulic state of the network is computed).
2. Water quality time step (time step used to track changes in water quality throughout the network.
3. Maximum trials allowed for hydraulic convergence (the maximum number of trials used to solve network hydraulics at each hydraulic time step).
4. Total normalised flow change for hydraulic convergence (convergence criterion that determines when a hydraulic solution has been reached; trials end when the sum of all flow changes from the previous solution divided by the total flow in all links is less than this value).
5. Maximum flow change for hydraulic convergence (additional convergence criteria that determines when a hydraulic solution has been reached; the largest absolute flow change between the current and previous solutions needs to be less than this value).
6. Maximum head loss error for hydraulic convergence (additional convergence criteria that determines when a hydraulic solution has been reached; the difference between the computed head loss and the difference between nodal heads across each link needs to be less than this value).
7. Frequency of hydraulic status checks (the number of solution trials that pass during hydraulic balancing before the status of links connected to tanks are updated).
8. Maximum trials for status checking (the number of solution trials after which periodic status checks are discontinued and checks are instead made only after convergence is achieved).
9. Accuracy level where solution damping begins (accuracy value at which solution damping and status checks on Pressure Reducing Valves (PRVs) and Pressure Sustaining Valves (PSVs) should begin).
10. Water quality tolerance (the difference in water quality level below which it is assumed that the quality of two parcels of water is the same).

Additionally, EPANET provides the following analysis statistics:

1. Number of hydraulic iterations taken.
2. Largest head loss error for links.
3. Cumulative water quality mass balance ratio.

VI. Management of data

The management of data needs two main elements. On the one hand, technical procedures ensure that the data is in good condition across the organisation, but it also requires that the people across the organisation is concerned about the management of data. On the other hand, next defined elements are required for a full management of data across the organisation.

VI.1. Data Inventory

The data inventory is a technical resource that can be implemented using anything from very simple tools (spreadsheet) to global structured tools able to manage the lineage of the different data.

It is a repository where all the data assets have to be compiled and identified. It also requires that these assets are prioritized according to the relevance for the organisation. Due to the fact that access to data is a critical factor for a successful management, an identity management solution should be implemented in order to ensure the correct access to the information by the granted stakeholders. Therefore, every asset registered has to include also the people able to access the data asset and what kind of permissions are granted.

VI.2. Data dictionary

The Data dictionary is another repository, depending or connected into the data inventory for the compilation of all the fields/properties of the data assets. It compiles not only what data asset the element belongs to but also the type of data in the field, and any restrictions/conditions of the values included. It helps in the design of new data sources across the organisation to find out information which is already in the organisation to avoid replications of data sources and the inherent inconsistencies.

VI.3. Data management procedures

In order to maintain proper management of the data across the organisation it is necessary to create, approve, and implement a group of procedures to manage the data assets. Otherwise these resources will not be adequately maintained and basically, they would turn in a waste of resources and an unbearable bureaucracy.

Below there is a generic description that has to be adapted to the particular needs of any organization. The initiative of smart data models has implemented part of them for their own use.

VI.4. Dataset registration process.

We are in the previous step before a department of the organization decides to create (or not) a new data set. How should we proceed? Following the detailed scheme below:

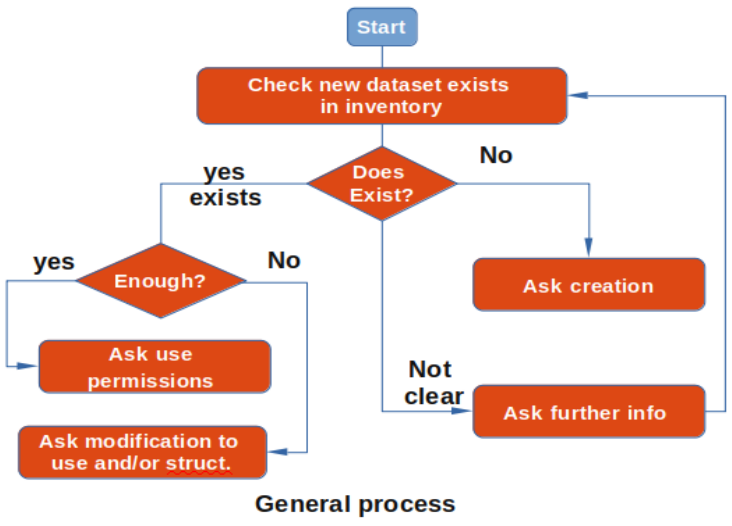

Figure 16: Dataset registration process

The general process considers the need of a user to access certain data within the organization. Due to the restrictions the user is not aware about its potential existence and therefore the first step is to find out the availability or the lack of these data within the inventory of data assets.

If the dataset exists and its structure meets the needs of the user, then the process ends by granting the permissions to access it. Sometimes a good precursor of the required data is available and with some modifications it could fulfil the needs. The process ends by asking those modifications and granting access to the user.

It is also possible that it is not so clear what are the demands in terms of data access from the user and accordingly further information would be requested before making any decision.

Finally, if the requested data does not exist a request for their creation is sent to the responsible person managing the data sources.

VI.5. Unsubscribe process

The unsubscribe process removes a data resource from the inventory. Usually because it is going to be replaced, but it does not impact on the process. Firstly, the process urges to communicate to the existing users of the data resource with the intention of removing the resource. Once gathered the issues, if there are no issues it is stored and finally removed from the inventory. In case there are some inconveniences, they are assessed and depending on the result can be rejected or done with a process for the removal before the final storage and final removal from the inventory.

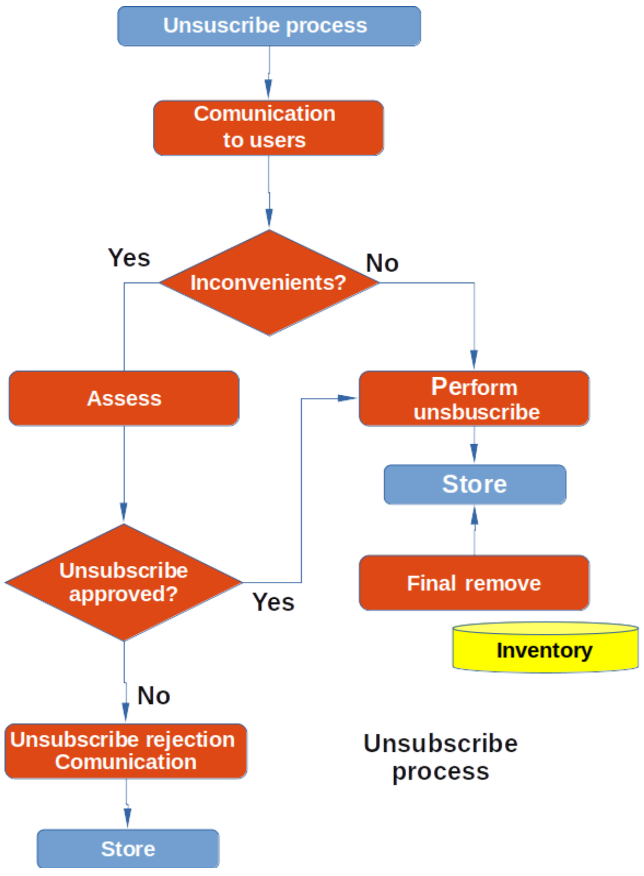

Figure 17: Unsubscribe process

VI.6. Evaluation process of changes to dataset

The change request is the process by which the structure of a data resource is reorganised (without complete backwards compatibility of data). A meeting or equivalent method has to be held with the existing users of the resource. As a consequence, a document with the input from the technical, organizational and point of view is generated including the availability of resources to face the change. If it is approved, then the plan is implemented, and the data resource is changed accordingly. In case the request is rejected, the changes plan is stored and registered for future needs.

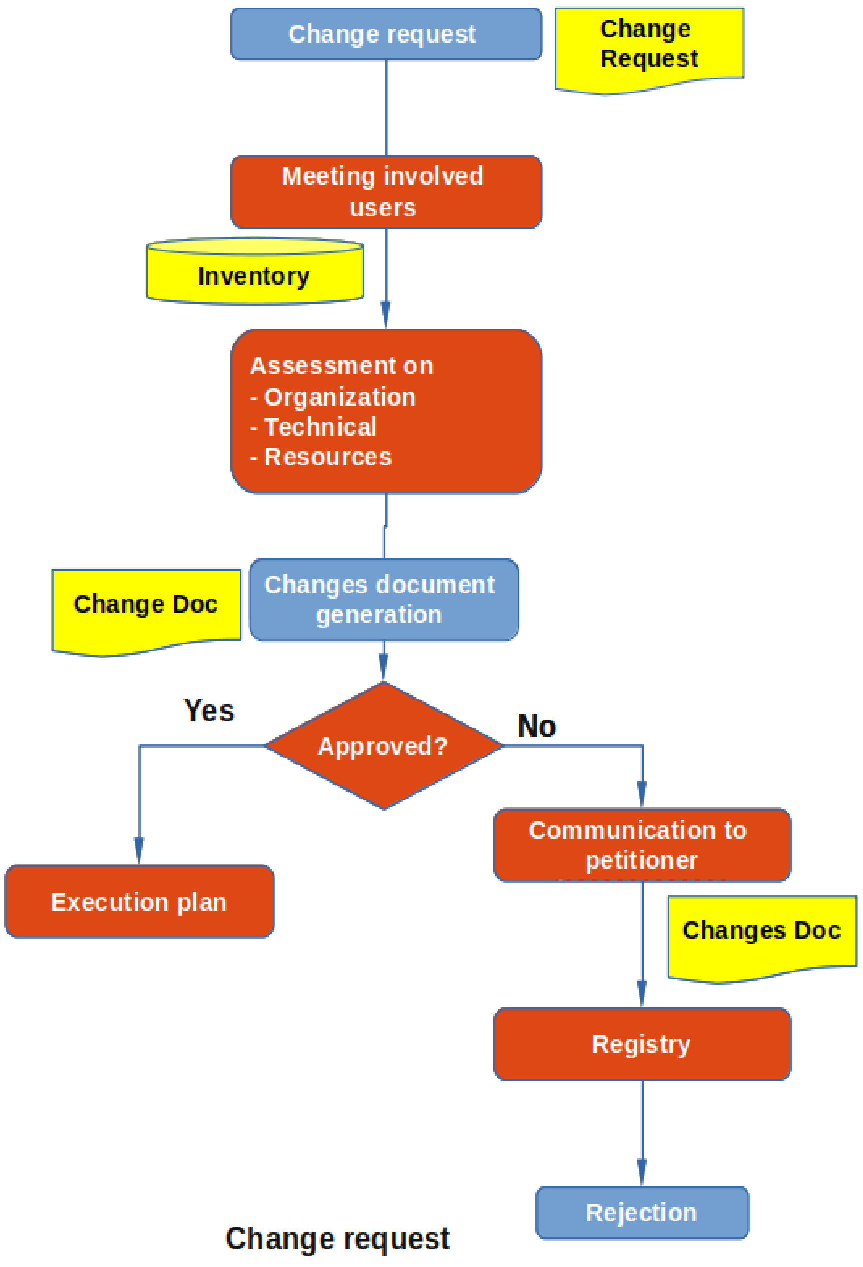

Figure 18: Evaluation process of changes to dataset

VI.7. Data registration process for new applications

A new application is a generator of new data assets. Depending on the purchase process and in the features of the application it has to be approached differently. Case it is a SaaS solution it has to be analysed the availability to access the business data within the service (including costs and resources required). If the data are not accessible the recommendation is to reject the application.

If it is accessible it has to be determined to what extent is configurable in terms of data, in order to make it compatible with the existing data assets. If there is not a possibility to change then directly store the structure into the data inventory. In case it is adaptable the recommendation is to include the change before putting into production the application.

If the data is contracted as a service (without deep integration into the business logic) it has to agree with the provider, the rights to access the data and to gather the rights to make it public with an open license.

Finally, if the application is in development, one of the requirements for the design has to be the compatibility with the existing data assets in the inventory. Consequently, once developed the data assets will have to be created in the inventory through the creation process.

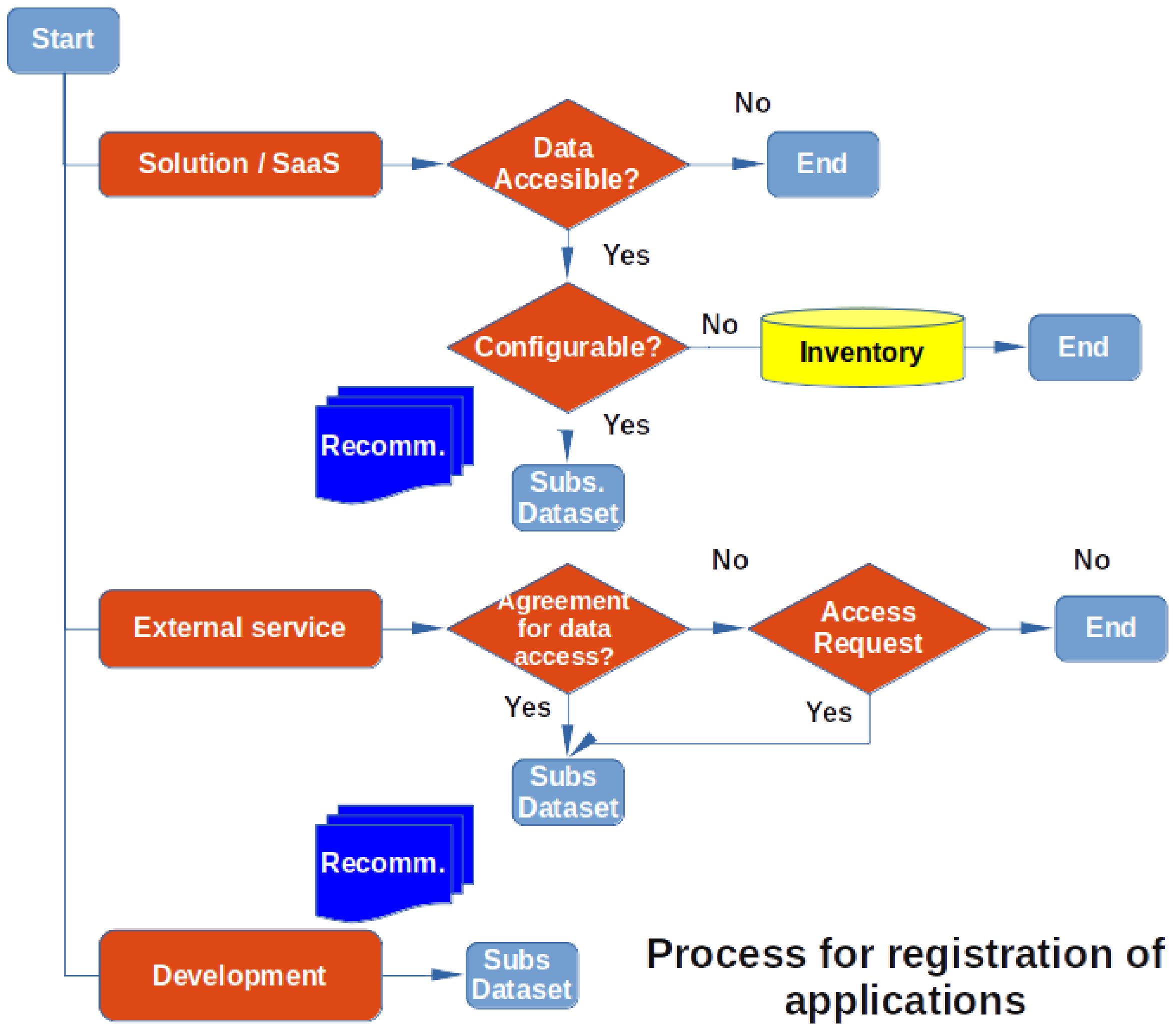

Figure 19: Data registration process for new applications

VI.8. Process Assignment of permits (data consumption)

New permissions on a data asset have to be granted once the identity of the person is confirmed and after permission of the data owner. The access permission to any data assets has to be maintained and included into the inventory database. Rejections are also required to be stored for security reasons.

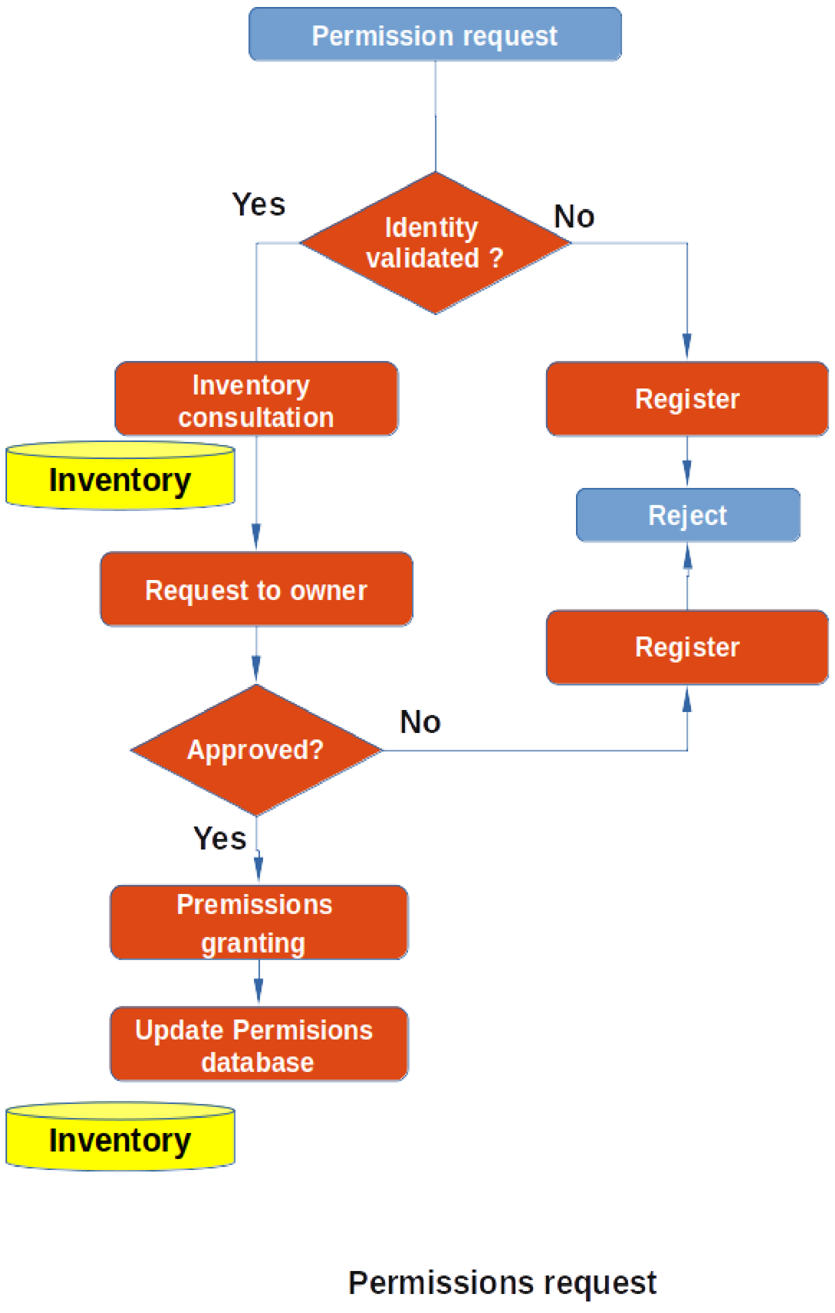

Figure 20: Process Assignment of permits (data consumption)

VI.9. Process Creation of new dataset.

The creation of a new data asset to be controlled has to be a restrictive process. Otherwise it could impose a bureaucracy to the organisation that impedes the benefits of the data asset control.

An organization can only control a small part of their data assets with specific procedures. The creation process applies to those data assets that have a priority which entitles them to be controlled and managed.

First step is to gather the needs and impacts on the potential users. Once agreed it has to be assessed the privacy and security conditions of the new data asset and stored in the inventory database. Additionally, it has to be analysed the actual impact on other users (based on the requirements gathered previously) and to proceed to the technical design. Once implemented the users should validate implementation before entering into production mode.

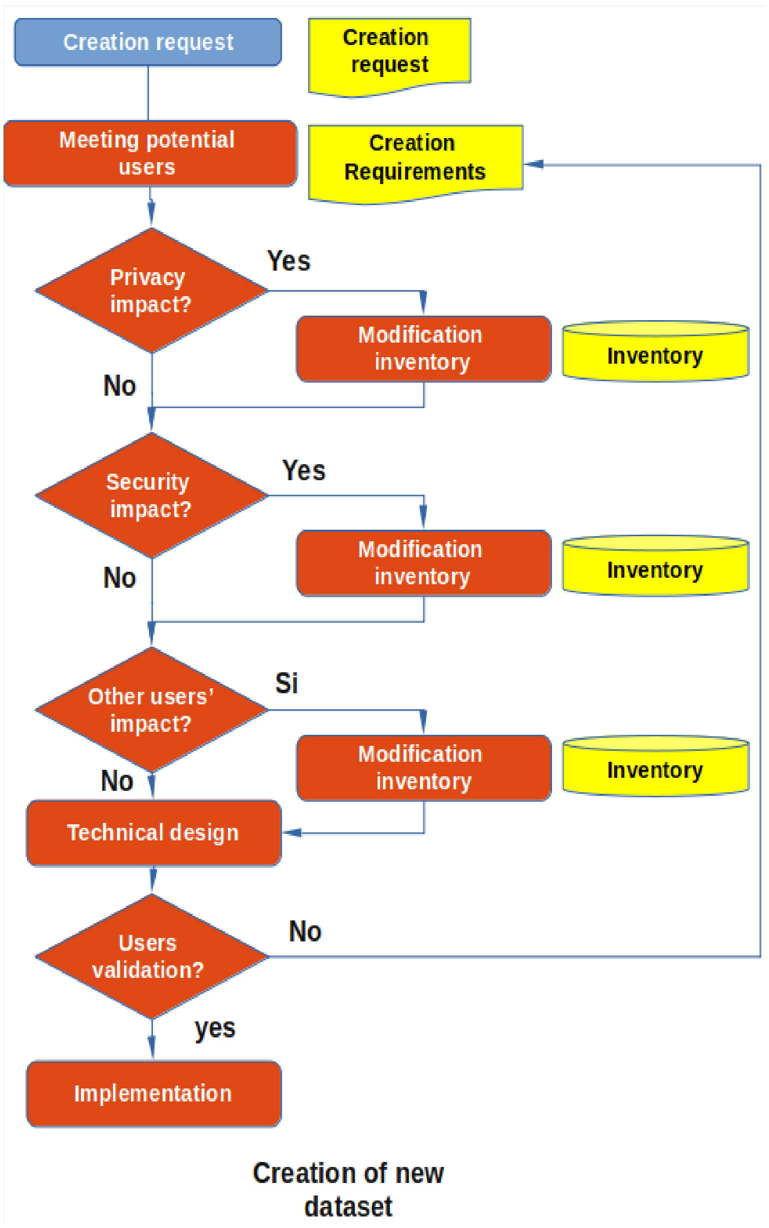

Figure 21: Process of creation a new dataset

VII. EU Added Value

We can understand the EU added value for the project as the value resulting from an EU project which is additional to the value that would have been created by individual states members alone. This means that there are several areas of interest to cover this added value:

- Networking, the project has involved the participation of different EU members and external stakeholders from Tunisia and India in the definition of the proper Smart Data Models to be used to share the Water information. The adoption of ETSI NGSI-LD standard as well as the FIWARE Technology have develop a research network to interact at pan-European level focussed on the results of the project.
- This collaboration in the definition of the ETSI standard as well as the Smart Data Models to be used in the standard have consequently facilitated the excellence and capacity building of the European partners involved on. This is basically because they are at the forefront of technology development.
- The collaboration of the FIWARE Foundation and specially the FIWARE Community provides us the opportunity to increase the visibility of the project. Additionally, this visibility, beyond EU countries, increase the geographic scope of the partners involved in the creation of the Fiware4Water Reference Architectures given the advantage offered by FIWARE Marketplace to develop business beyond EU countries.
- It is well-known that one of the problem in the Water Sector, and even other vertical sectors, is the leak of data harmonization and data management. The main purpose of the Fiware4Water Reference Architecture (F4W-RA) is to offer an architecture, free to use, that allows open interoperability of services using a standard context information representation easily understandable by human and services. this approach will lead to a harmonisation of water management services at both European and pan-European level.
- Moreover, the use of open source components with open APIs and standards is well-known to lead a cost reduction of the services to be developed as well as prevent the redundancies in the Smart Water services due to all of them can share the same context information representation as well as the same open interfaces to access the data.

Conclusion and Perspectives

In this report, it is presented the reference architecture of the FIWARE4Water platform. This platform targets the integration of any legacy and existing water management system into a large-scale standard platform based on FIWARE Technology. The goal is to allow all water sector applications to run on a homogeneous infrastructure, utilizing standard data exchange models to represent the context information and using standard APIs to access and share the information.

Existing data modelling approaches typically address only parts of the aspects required on the different abstraction layers. Moreover, they are only focused on a very concrete use case and therefore, they are not standardised to any other broader solutions. That is the main reason explaining the important interoperability problems that water management applications are currently facing. Higher-level ontology is often not suitable for modelling context information due to their abstraction in the representation of the information. Nevertheless, the use of standard data models based on such defined ontologies can easily mitigate or even resolve these problems. This is the main purpose of the selection of the ETSI NGSI-LD API to access and manage the context information and the use of the FIWARE Smart Data Models aligned with ETSI ISG CIM and ETSI SAREF for representing of the context information. In a relevant use case, the integration with EPANET have required the definition of new data models, already available with an open license, capable to map accurately the needs of this management software.

Of course, different scenarios may use their own data models and representation of sensor information, therefore matching and transforming this context information will be necessary in FIWARE4Water through the use of specific FIWARE IoT Agents. In addition, different types of processing require different representations, especially when we are talking about Machine Learning, Artificial Intelligence and Big Data processing activities. This is something that will be covered in the subsequent activities of the WP2, especially inside Task 2.2.

In addition, an important challenge will be to assess the performance of the FIWARE4Water platform and demonstrate at large scale the technical feasibility of using FIWARE Technology and FIWARE Smart Data Models inside the water domain. Thus, it will demonstrate the generation of newer digital services using a reference architecture across the sector, enabling interoperability and data sharing. Hence, the idea is to evaluate them as the flexible digital open source solution of choice in a variety of diverse real-world applications covering a wide range of water challenges and contexts. These real-world contexts will be provided by the four (4) large scale demo cases of the FIWARE4Water project, which will be the living laboratories for the testing, validation and demonstration (WP4) of the Smart Water Apps developed and the Smart Water Devices customised during the project (WP3).

Moreover, modelling the Quality of Context Information (QCI) is an important aspect for all the smart context information applications that we will develop. Different quality aspects like precision, accuracy, temporal validation, completeness, outliers, null/empty values, geolocation correctness, duplicates, inconsistent replicated data, wrong codifications, and out of normalized data have been described in the literature, but the definitions for these terms vary significantly and no standard has yet emerged. Different approaches have been proposed to describe how this QCI can be integrated with different kinds of attributes inside the smart data models or the use of concrete CEP mechanisms automatically associated to the context information. Therefore, an important challenge for the FIWARE4Water platform will be to:

- determine how to model context information on the different levels to include this QCI, and
- define how to infer the resulting QCI and the resulting valid Context Information based on the higher-level context information together with the corresponding associated QCI serving as input.

By levels, we refer to the different actions to take in different layers of the architecture where we can actuate on the context information in order to calculate the corresponding QCI indicators, and ultimately allow or discard this context information because of its poor QCI indicator or indicators.

Furthermore, QCI will help users to succeed in managing their data. This is a progressively relevant aspect of the water management sector, in which the increasing size of the data collected, together with the rise of different data types, makes data management necessary for successful water management. FIWARE4Water project provides extensive examples and procedures to ensure a proper use of water data.

Cybersecurity is another fundamental aspect to be integrated in any aspects of the platforms following a security by design paradigm. Starting from the platform conception, building upon an open-source ecosystem raises challenges from the integration phase as code analysis has to be conducted for any software dependencies part of the platform, including one from third parties. Then the deployment phase, often built upon containerised and scalable environments requires special attention, transforming traditional DevOps software development organisation into DevSecOps one. Finally, traditional security tests have to be deployed at run time to maintain system security. This includes compliance to GDPR requirements.

Last but not least, the adoption of FIWARE technology and therefore, the use of ETSI NGSI-LD provides a good opportunity for the CNRS nanosensor (the former so called “PROTEUS sensor”) to be FIWARE compliant. FIWARE-ready IoT devices come with easy-to-install drivers and instructions that help to transform the measures they gather into context information, accessible to applications using the ETSI NGSI-LD standard. The main advantage of this approach is that it does not require the use of a Gateway API (a.k.a. FIWARE IoT Agent) to transform the legacy representation format and transport protocol into NGSI-LD. The communication can hence be directly sent to the proper FIWARE Context Broker. For this purpose, the FIWARE4Water partners, through WP2, will help CNRS to certify its CRNS nanosensor. Currently, this sensor is an innovation product at TRL5 level and the purpose at the end of the project is to reach TRL7 level. Therefore, from the point of view of the FIWARE Ecosystem, obtaining a "FIWARE-Ready IoT device" certification could open the possibility to access to the FIWARE Market in a preference position.

References

[1] Abella, A., Ortiz-de-Urbina-Criado, M., & De-Pablos-Heredero, C. (2019). Meloda 5: A metric to assess open data reusability. *El profesional de la información (EPI)*, *28*(6).

[2] JSON-LD (A JSON-based Serialization for Linked Data), v1.1, W3C Proposed Recommendation 07 May 2020, https://www.w3.org/TR/json-ld11.

[3] ETSI Context Information Management (CIM); NGSI-LD API, ETSI GS CIM 009 V1.2.2 (2020-02) https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.02.02_60/gs_CIM009v010202p.pdf

[4] Rossman, L.A. (2000) EPANET 2 Users Manual. EPA/600/R-00/057, U.S. Environmental Protection Agency, Cincinnati, OH.

[5] Mo Jamshidi, "From Large-Scale Systems to Cyber-Physical Systems," *Journal of Internet Technology*, vol. 12, no. 3 , pp. 367-374, May. 2011.

[6] A Brief Introduction to XACML, Sun Microsystem, Inc., 2003, https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html

[7] OpenAIRE Guidelines for Data Archives, OpenAIRE, 2015, licensed under Creative Commons Attribution 4.0 International. https://guidelines.openaire.eu/en/latest/data/index.html

[8] McKeever, S. et al.: A Context Quality Model to Support Transparent Reasoning with Uncertain Context, In Proceedings of 1st Workshop on Quality of Context (QuaCon), Stuttgart, Germany, 2009.

[9] Sheikh, K., Wegdam, M., van Sinderen, M. (2007). Middleware support for quality of context in pervasive context-aware systems. In: Proceedings of 5th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW). pp. 461-466.

[10] Manzoor, A., Truong, H.L., Dustdar, S. (2008). On the evaluation of quality of context. In: Proceedings of European Conference on Smart Sensing and Context (EuroSSC). pp. 140-153.

[11] Bu, Y., Gu, T., Tao, X., Li, J., Chen, S., Lu, J. (2006). Managing quality of context in pervasive computing. In: Proceedings of 6th International Conference on Quality Software (QSIC). pp. 193-200.

[12] Lu, Y., Kumar, J., Collier, N. O., Krishna, B., Langston, M. A. (2018). Detecting outliers in streaming time series data from ARM distributed sensors. In: Proceedings of Detecting outliers in streaming time series data from ARM distributed sensors conference.

[13] Open Water Analytics. "OWA-EPANET Toolkit 2.2." http://wateranalytics.org/EPANET/_options_page.html (accessed 11th June 2020).