PREDIS

Deliverable 7.7

Innovative data handling, processing, fusion, and decision framework technologies used for assessing cemented waste package safety and maintenance strategies

21.2.2024 Version 1.2 Final

Dissemination level: Public

Joonas Linnosmaa
VTT Technical Research Centre of Finland
Visiokatu 4, Tampere, Finland

email: joonas.linnosmaa@vtt.fi

This project has received funding from the Euratom research and training programme 2019-2020 under grant agreement No 945098.

Project acronym PREDIS	Project title PRE-DISposal management of radioactive waste		Grant agreement No. 945098
Deliverable No. D7.7	Deliverable title Innovative data handling, processing, fusion, and decision framework technologies used for assessing cemented waste package safety and maintenance strategies		Version 1.2
Type Report	Dissemination level Public		Due date M42
Lead beneficiary VTT			WP No. 7
Main author Joonas Linnosmaa (VTT)	Reviewed by Ernst Niederleithinger (BAM), WP7 Lead	Accepted by Maria Oksa (VTT), Coordinator	
Contributing authors Teemu Mätäsniemi (VTT), Tero Jokinen (VTT), Petri Kaarmila (VTT), Tuula Hakkarainen (VTT), Enrico Botta (ANN), Tom-Robert Bryntesen (IFE), Réka Szőke (IFE)			Pages 77

Abstract

This work is a part of the European project, PREDIS, on pre-disposal treatment and management of low-level waste and intermediate-level waste, The state of the art report (D7.1) and the gap analysis (WP2) have identified the need for more research on adequate and industrially mature solutions (practices, data governance, technology, and tools) for monitoring the packages and supporting decisions in the preparation, handling, and long-term interim storage of low-level/intermediate-level cemented waste.

In the frame of task 7.5, a data management framework was set up taking into account all information gained from read sensor data as well as processed and analysed data. These were collected and stored to enable and develop the decision-making process for cemented waste packages in pre-disposal storages. The work aimed to study and demonstrate data management practices, which help the end-users (waste, and data management organizations) to increase their organizational maturity. This was done by identifying the most relevant business processes and specifying possible technical enabler systems.

The most relevant processes identified were the measurement process, system analysis process, information management process, and decision management process. Accompanying them, three enabler systems were specified, developed, and demonstrated for data platforms, and decision frameworks. System enablers were integrated into a data management framework, which manages integrated data from sensing to decision-making. This working report focuses on reporting the development work of the data platform, and the decision framework.

This work is strongly connected to task 7.3 “Innovative integrity testing and monitoring techniques” (reported in D7.3) and task 7.4 “Digital twin” (reported in D7.5) which is on simulation and digital twins.

Coordinator Contact

Maria Oksa
VTT Technical Research Centre of Finland Ltd
Kivimiehentie 3, Espoo / P.O. Box 1000, 02044 VTT, Finland
E-mail: maria.oksa.@vtt.fi
Tel: +358 50 5365 844

Acknowledgment

This project has received funding from the Euratom research and training programme 2019-2020 under grant agreement No 945098.

TABLE OF CONTENTS

List of Abbreviations

Term	Definition
3D	Three-dimensional
ACID	Atomicity, Consistency, Isolation and Durability
ACL	Access Control List
ADC	Analog to Digital Converter
AE	Acoustic Emission
AI	Artificial Intelligence
ALARA	As Low As Reasonably Achievable
ANN	Ansaldo Nucleare SpA
API	Application Programming Interface
ARM	Azure Resource Manager
ASR	Alkali-Silica Reaction
AWS	Amazon Web Services [cloud]
BASE	Basic Availability, Soft state and Eventual Consistency
CCOM	Common Collaborative Object Model
CI/CD	Continuous Integration and Continuous Deployment
CLI	Command Line Interface
CPU	Central Processing Unit
CRDS	Cavity Ring-Down Spectroscopy
CRIS	Common Relational Information Schema
CRM	Customer Relationship Management
DAU	Data Acquisition Unit
DBA	Database Administrator
DIC	Digital Image Correlation
DLM	Data Lifecycle Management
EJP	European Joint Programme
ERP	Enterprise Resource Planning
EU	European Union
HLW	High-Level [radioactive] Waste
HMI	Human-Machine Interface
HTML	Hypertext Markup Language
Http	Hypertext Transfer Protocol
HW	Hardware
I/O	Input/Output
IAM	Identity and Access Management
IFE	Institutt for energiteknikk
ILW	Intermediate-Level [radioactive] Waste

Term	Definition
IoT	Internet of Things
IT	Information Technology
LLW	Low-Level [radioactive] Waste
LoRa	Longe Range Radio
MAC	Media Access Control
MES	Manufacturing Execution Systems
MIT	Massachusetts Institute of Technology
MLW	Medium-Level [radioactive] Waste
MQTT	Message Queuing Telemetry Transport
Mu-Tom	Muon Tomography
MVC	Model-View-Controller
MVT	Model-View-Template
NaN	Not a Number
NDE	Non-Destructive Evaluation
NDT	Non-Destructive Testing
NoSQL	Not Only SQL / Non-SQL
O&M	Operations and Maintenance
OLAP	Online Analytical Processing
OM	Ontology of Unit of Measure
OSA-CBM	Open Systems Architecture for Condition Based Maintenance
OSA-EAI	Open Systems Architecture for Enterprise Application Integration
OWA	Open-World Assumption
P	Pressure
PaaS	Platform as a Service
PROV-O	Provenance Ontology
RAM	Random-Access Memory
RBAC	Role-Base Access Control
RDBMS	Relational Database Management System
RFID	Radio Frequency Identification
RH	Relative Humidity
RSSI	Received Signal Strength Indicator
RTOS	Real Time Operating System
SOSA	Sensor, Observation, Sample and Actuator [ontology]
SQL	Structured Query Language
SRA	Strategic Research Agenda
SSD	Solid-State Drive
SSN	Semantic Sensor Network [ontology]
SW	Software

Term	Definition
T	Temperature
URL	Uniform Resource Locator
UTF	Unicode Transformation Format
VLLW	Very Low-Level [radioactive] Waste
VTT	Teknologian tutkimuskeskus VTT Oy
W3C	World Wide Web Consortium
WP	Work Package
WSGI	Web Server Gateway Interface
WSN	Wireless Sensor Network

1 Introduction

Efficient and safe management of radioactive waste is of utmost importance. By understanding the nature of the waste, engaging with the stakeholder community, investing in research and development, conducting risk assessments, and implementing sustainable practices, we can achieve safe management of radioactive waste, thereby protecting both present and future generations. Solid and liquid radioactive waste is continually generated by the nuclear industry and institutional producers, such as healthcare and research facilities. Such waste must be appropriately conditioned to be acceptable for storage and eventual disposal. Among the possible conditioning processes, encapsulation of waste using an industrial cementitious grout is a method currently implemented by many EU member states.

Medium to long-term storage, a strategy indicated in the National Programmes of many European countries to deal with delays in available disposal solutions, necessitates the monitoring of waste packages for potential degradation phenomena. Managing any arising issues prior to transport to the final repositories is crucial. Innovation in degradation prevention and early detection through advanced monitoring systems based on innovative techniques offers significant opportunities. These advancements can lead to improved storage operations, reduced costs, increased safety, and a better understanding of the waste's characteristics prior to final disposal. This approach aligns with the goals of all member states, ensuring a more efficient and secure handling of radioactive waste.

Project description

The PREDIS project, titled "Predisposal Management of Radioactive Waste", builds upon earlier work on international roadmaps and strategic research agendas. However, it strives to make significant advances by fostering a more holistic perspective on the pre-disposal needs and priorities for the future. The project structure was established to address four waste streams identified as priority areas by the end-user community within the scope of the European Joint Programme (EJP) in Radioactive Waste Management. The project's primary aim is to bring measurable benefits to the Member States and the nuclear waste community. This includes the further development and increase in the Technological Readiness Level of treatment and conditioning methodologies for wastes for which no adequate or industrially mature solutions are currently available. Specific areas of focus include metallic material (WP4), liquid organic waste (WP5), solid organic waste (WP6), and innovations in cemented waste handling and pre-disposal storage (WP7), their rough relationships can be seen in Figure 1. This report focuses on the work done in WP7.

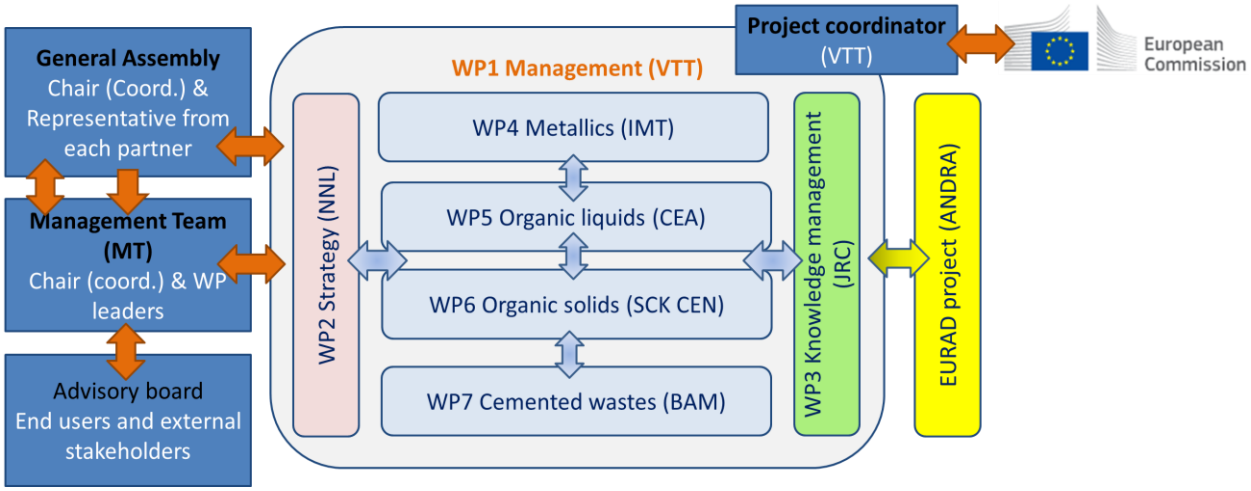


Figure 1. PREDIS project structure.

Work Package 7

The PREDIS WP7, titled “Innovations in Cemented Waste Handling and Pre-Disposal Storage”, aims to innovate in the areas of degradation prevention and early detection using advanced monitoring systems based on innovative techniques. WP7 offers significant opportunities for improving storage operations, reducing costs, increasing safety, and enhancing the understanding of waste characteristics prior to final disposal.

WP7 has the following objectives:

- Compile Information: Gather and analyse information about the state-of-the-art methods and procedures for cemented waste management, with a specific focus on monitoring during preparation, handling, and long-term storage.
- Quality Assurance and Monitoring: Identify, evaluate, and demonstrate store and package quality assurance (mainly based on non-destructive methods) and monitoring technologies.
- Modelling Capability: Demonstrate the capability of geochemical and chemo-mechanical models to describe the chemical and mineralogical evolution of packages, and to validate their suitability for disposal.
- Digital Twin Technology: Develop, adapt, and demonstrate digital twin technology, methods for data handling, and an overall digital decision framework.
- Increased Store Automation: Identify opportunities for increasing store automation to reduce human exposure to radiation.
- Package Improvement: Identify options for post-treatment of packages and potential approaches to improve package design, construction, and maintenance.

While the first three objectives have led to planning and carrying out actual demonstrations, the latter three are limited to desk studies due to time and budget constraints. The common goal of WP7 tasks is to develop a concept and a prototype for a condition monitoring system. More specifically this report focuses on Task 5 of WP7.

Task 7.5

This document reports the results of Task 7.5, which had the goal to produce:

- Conceptual Model for Data Handling and Storage: Provide a conceptual model for secure and persistent data handling and storage. This includes both measured data (Task 3) and simulated data (Task 4).
- Translation of NDE and Monitoring Data: Develop models and methods to translate Non-Destructive Evaluation (NDE) and monitoring data into engineering parameters.
- Fusion of Multi-Method Monitoring: Develop approaches for the fusion of multi-method monitoring and other data. This will enable obtaining adequate input for the decision framework.
- Data Integrity Plan: Implement a plan to ensure the integrity of the data throughout the project's lifecycle.
- Database and Software Prototype: Provide a database and software prototype for demonstration purposes. This will serve as a tangible example of the project's capabilities and potential applications.

The task collectively contributes also to the overarching goal of innovating in the areas of degradation prevention, early detection, and efficient handling of cemented waste. Figure 2 presents the relationship of Task 7.5 to Task 7.3 (Innovative integrity testing and monitoring techniques) and Task

7.4 (Digital Twin). Also, the subtasks of Task 7.5 and their focus areas are visualized. Within the task we are calling this concept **data management framework.**

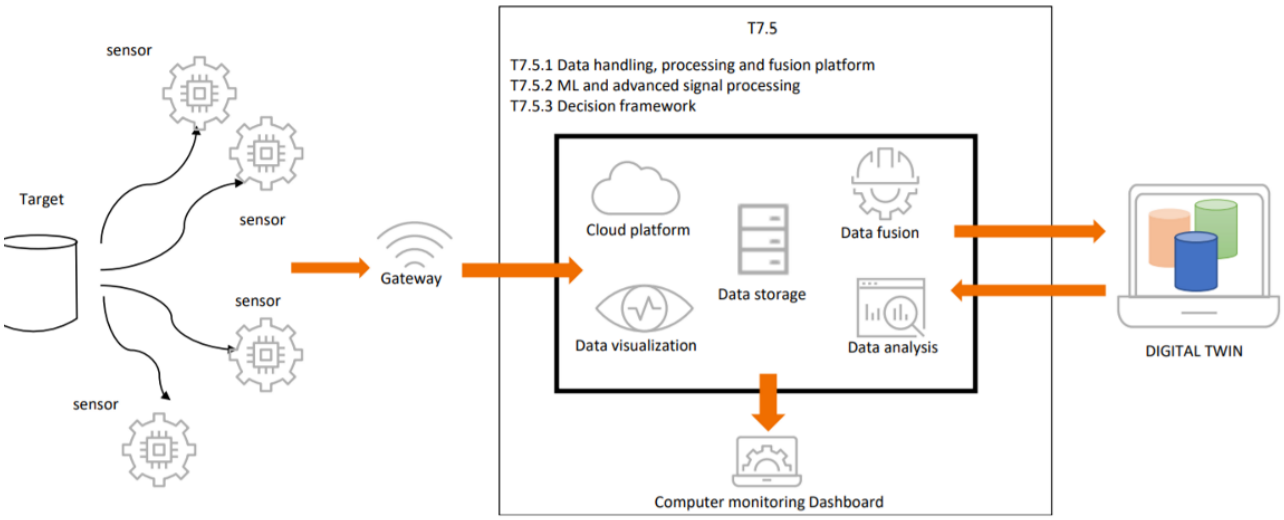


Figure 2. Scope of Task 7.5: Data handling, processing, and fusion.

Sub-tasks 7.5.1, 7.5.2 and 7.5.3

Task 7.5 was further divided into three subtasks:

- The first Subtask 7.5.1 involves developing a platform with the goal of fostering seamless connections between devices and sensors from various manufacturers, including those used in the monitoring system. A configurable Human Machine Interface (HMI) was planned to be implemented into the platform, providing real-time status of field sensors, time trends, alarm thresholds, and more, with possible access to third parties such as safety authorities and fire brigades. A "dashboard concept" will be developed for the platform, enabling visualisation of model predictions of event evolution to be assessed alongside real-time sensor measurements. Additionally, a reference database will be planned and conceptualized, possibly containing waste-condition information from various stages throughout the waste lifecycle, for example acceptance, conditioning, and pre-disposal storage.
- The second Subtask 7.5.2 was planned to focus on implementing more general data processing techniques. However, because of the challenges of producing measurement dataset from the project, and the need for further planning in the final implementation, this task focuses more on the selected general approach implementing activities such as communication, system integration and the administration of the various platforms used for the development work, such as Microsoft Azure and InfluxDB.
- The third Subtask 7.5.3 focuses on the decision-support framework. This framework will utilize the available data, computed results, predictions from the digital twin, and expert knowledge, converting this information into end-user digestible visualization, recommendations, and (semi-)automated decisions for the end user. A prototype decision support platform will be implemented to integrate data from monitoring with the scheduling of preventive maintenance actions, optimizing maintenance strategies, and safety assessments. Strategies will also be developed for optimizing package arrays about store requirements using the digital decision framework.

Together, these sub-tasks form a comprehensive approach to handling, processing, and utilizing data in the context of cemented waste management, leveraging modern technologies and

methodologies to enhance efficiency, safety, and decision-making, thus building a concept for a common data management framework.

Connections to other PREDIS tasks

The interrelations of the tasks create information dependencies and technological harmonization and integration requirements for the Task 7.5 as presented in Figure 3. The data synthesis and technology integrations cover both raw data and meta data.

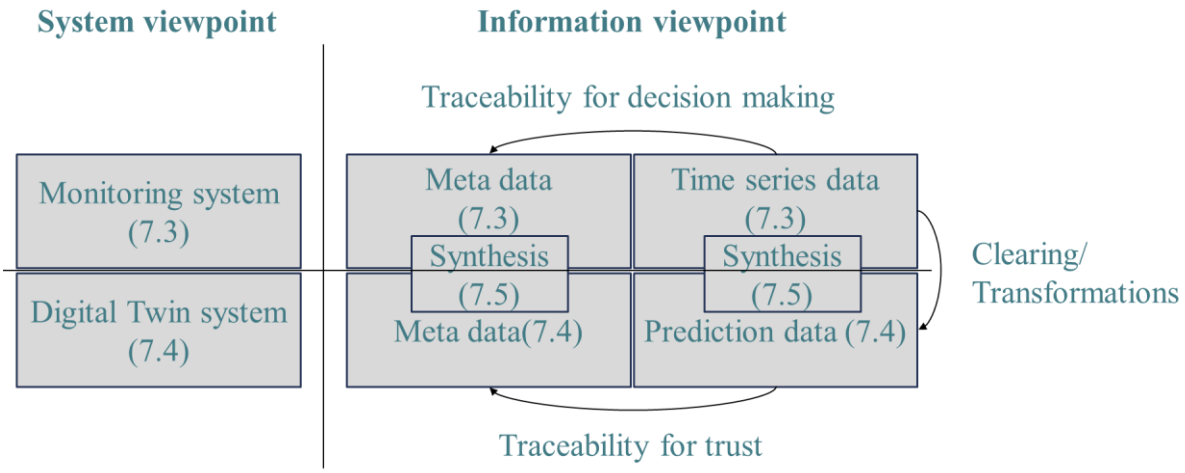


Figure 3. Information dependencies and integration requirements.

As shown in Figure 3, task 7.5 is strictly connected with other WP7 tasks. In the following we give some context and reasons for these links:

- It receives input information and guidance from Task 7.2, devoted to:
  - Compile a state-of-the-art report concerning the packaging, storage, and monitoring strategies for cemented waste packages in Europe
  - Identify a reference container/package type and a list of reference package evolution and degradation scenarios to be used
  - Provide necessary data for the strategic environmental and impact assessments
- It receives data (measured and numerical) from Task 7.3, devoted to
  - Select relevant conventional and innovative NDE/monitoring techniques and adapt them for use under typical storage conditions, for implementation into individual waste packages, for wireless data transmission and for wireless energy supply
  - Develop an instrumentation setup for use in Task 6 that incorporates a combination of techniques, including both sensors embedded in individual waste packages and measurement systems integrated into the repository
  - Test the relevant technologies at UJV using full-size package mock-ups that incorporate real cementation technology
- It provides methods for demonstration in Task 7.6, devoted to:
  - Demonstrate that the technologies, methods, and models developed and identified in Tasks 7.2 to 7.5 can be used in a nuclear environment
  - Test and verify the performance of the selected technologies, developed prototypes, and models by the deployment of an instrumented package at an end-user facility, possibly within a store environment
  - It receives information on treatment and other mitigation options from Task 7.6.
- Interfacing with Task 7.4 focusing on digital twin model, it is also required to allow access to a selected dataset and return data for predictions.

- Task 7.4 provides simulated data of waste form evolution for Task 7.3 and supports sub-task 7.5.3 with data and expertise needed for the decision-making framework.
- It provides requirements for the data repository.

2 Condition monitoring and maintenance

2.1 Why condition monitoring?

Condition monitoring systems are essential tools in modern industrial operations, providing real-time insights into the performance, health, and efficiency of machinery, equipment, and processes. These systems play a vital role in predictive maintenance, safety, and optimization, allowing organizations to make informed decisions based on accurate and timely data.

What is Condition Monitoring? Condition monitoring refers to the continuous or periodic observation and analysis of the operating parameters of a system. It involves the use of various sensors, instruments, and software to collect, process, and interpret data related to temperature, vibration, pressure, humidity, and other physical properties. The goal is to detect any changes or anomalies that may indicate potential problems, wear, or failure (Rao, B. K. N., 1996).

Why is Condition Monitoring Important?

- Predictive Maintenance: By identifying early signs of wear or malfunction, condition monitoring enables timely maintenance and repairs, preventing unexpected breakdowns and minimizing downtime.
- Safety: Monitoring the condition of equipment and processes helps ensure that they are operating within safe parameters, reducing the risk of accidents and hazards.
- Efficiency: Through continuous analysis of performance data, condition monitoring can identify areas for optimization, leading to energy savings and increased productivity.
- Cost Savings: By enabling proactive maintenance and reducing the need for emergency repairs, condition monitoring can significantly reduce overall maintenance costs.
- Compliance: In regulated industries, condition monitoring helps ensure that equipment and processes comply with legal and environmental standards. (Randall, R. B., 2011)

Key Components of Condition Monitoring Systems:

- Sensors: Various types of sensors are used to measure physical properties such as temperature, vibration, pressure, etc.
- Data Acquisition: Devices and software that collect and transmit data from the sensors to a central system.
- Data Analysis: Software tools that analyse the collected data, applying algorithms and statistical methods to detect patterns and anomalies.
- Human-Machine Interface (HMI): A user-friendly interface that allows operators to view and interact with the data, often including visualizations, alarms, and reports.
- Integration with Other Systems: Many condition monitoring systems can be integrated with other enterprise systems, such as Enterprise Resource Planning (ERP) or Manufacturing Execution Systems (MES), for holistic management. (Jardine et al., 2006)

Condition monitoring systems are a cornerstone of modern industrial practice, providing essential insights into the health and performance of machinery and processes. Whether in the context of cemented waste management, manufacturing, energy production, or other sectors, these systems enable organizations to operate more safely, efficiently, and cost-effectively. By leveraging advanced sensors, data analytics, and integration capabilities, condition monitoring paves the way for

intelligent decision-making and proactive management in today's complex industrial landscape. (Mohanty, 2015).

2.2 Condition monitoring techniques for radioactive waste

In cemented Radioactive waste, the essential parameters to be monitored according to the survey reported (PREDIS Deliverable 7.1, 2021-04-14 version 1.1):

- Identification: Determine the identifiers for the barrels of radioactive waste
- Metal corrosion
- Chemical reactions
- Leakage
- Lifting feature deformation
- Swelling
- Cracks
- Condensation
- Contamination of waste barrels
- Fissile content

Condition monitoring techniques for cemented radioactive waste that can be used in this project have been reported (PREDIS Work Package 7 Milestone MS53 – M7.5 – Month 36 Internal Report):

- Long Range Radio (LoRa) communication technology: The use of LoRa nodes allows to assign unique identifiers to barrels of radioactive waste and at the same time periodically collect their radiation data. LoRa gateways control data traffic and forward data to cloud services and thus enable remote access to data.
- Acoustic emission (AE) monitors acoustic emissions or high-frequency signals and can be used for non-destructive monitoring of possible cracking of concrete. E.g., Alkali-silica reaction (ASR) and the subsequent swelling of the gel-like product can cause cracking in concrete.
- The non-contact air-coupled ultrasonic sensors can be used to provide for drums inspections such as measuring circumferences to detect drum swelling and discontinuity defects such as cracks or corrosion cavities.
- Muon Tomography technique (Mu-Tom) enable a non-destructive approach to investigate the internal unknown composition of cemented drums.
- Neutron and gamma monitoring system with smart electronics
- Customized RFID for embedded sensors inside waste drums

Similar techniques have been used in other applications domains. The following is a list of the most typical condition monitoring techniques that are used to assess the condition of machines, devices, or systems, which can be helpful for the monitoring of radioactive waste too:

- Thermography: Infrared cameras are used to monitor the temperature variations of devices and thus detect, for example, overheating of components.
- Ultrasound Testing: Monitoring high-frequency sound waves to identify problems such as leaks, electrical faults or bearing problems.
- Visual Inspection: Identifying visually signs of wear, leaks or corrosion of components.
- Acoustic Emission Testing: Monitoring acoustic emissions or high-frequency signals and detecting material degradation or structural defects in components.
- Performance monitoring: Monitoring the performance parameters of machines and systems and detecting possible deviations from the expected operating behaviour.

- Remote monitoring and IoT: Enables remote monitoring and analysis of the condition of machines from a centralized location by utilizing sensors, IoT devices and connections to collect real-time information from the devices.

These techniques can be combined for different uses and requirements. Condition monitoring enables maintenance strategies, e.g., condition-based maintenance and predictive maintenance.

Some condition monitoring techniques have their own standards, for example:

- ISO 13373-1 Condition monitoring and diagnostics of machines — Vibration condition monitoring — Part 1: General procedures
- ISO 13373-2 Condition monitoring and diagnostics of machines — Vibration condition monitoring — Part 2: Processing, analysis and presentation of vibration data
- ISO 14830-1:2019 Condition monitoring and diagnostics of machine systems — Tribology-based monitoring and diagnostics — Part 1: General requirements and guideline
- ISO 18434-1 Condition monitoring and diagnostics of machines — Thermography — Part 1: General procedures
- ISO 17359:2019 Condition monitoring and diagnostics of machines — General guidelines

2.3 Functional specification of condition monitoring system

This section introduces general and possible functionalities for a machine condition monitoring, but the ideas are also adaptable for the architecture of novel condition monitoring system in cemented waste management. The generic descriptions of the functionalities are presented here, but later, in Chapter 3.1, they are allocated to the different subsystems of the data management framework.

ISO 13374-1 (2003) establishes general guidelines for software specifications to allow machine condition monitoring data and information to be processed, communicated, and displayed without platform-specific or hardware-specific protocols. This kind of architecture should operate in plug-and-play fashion and support pro-active operation, maintenance and discovering root causes and severities of possible faults. Information flows from data acquisition to advisory generation are in the scope. The guideline introduces six distinct and layered data-processing blocks for which more detailed requirements are stated in ISO 13374-2 (2008). A software module implementation may include functionality of one or several data-processing blocks. The blocks are introduced as follows.

The **data acquisition block** is for identification of measurement locations, sensor configurations, signal to parameter conversions and configuring plant/asset/part hierarchies. The block may read data from sensors or get a manual or automatic feed to collect, digitalize and consolidate data to records into a repository.

The **data manipulation block** is for signal analysis and virtual sensor descriptors. The block can convert raw data to meaningful features which characterize the feature of interest. Typical processing algorithms are filtering, windowing, averaging, normalizing and time-frequency transformations.

The **state detection block** is defined for baseline profiles, limit values, abnormality search or baseline comparison and anomaly detection. The block generates state indicators with degree of abnormality. Also, statistical measures such as Weibull and Gaussian distributions may be generated. All generated assessments must be based on operational context and are sensitive to the current operational state.

The **health assessment block** generates alerts and alarms, includes diagnostic analyses for faults, possible failures with probabilities and estimates machine health index based on human or agent expertise. In addition, evidence and explanation information may be included in reporting and visualizations.

The **prognostic assessment block** predicts the expected life of the machine or fault conditions by determining future health states and failure modes based on the current situation and projected usage loads. The future usage load may be forecasted with operational models or historical data.

Lastly, the **advisory generation block** supports the optimization of the life of the equipment and recommends alternative actions for future operation, maintenance, or strategic activities. Also, capability forecast assessment or operational profile modification is of interest. The advisory generation block as a decision support module considers constraints such as the operational usage and maintenance history, current and future mission profiles, high-level objectives, and other resource constraints.

In addition, this software specification is based on a three-layer architecture by separating data archiving, processing logic, and information presentation layers.

MIMOSA™ is a not-for-profit industry trade association that develops and encourages the adoption of open, supplier-neutral standards enabling physical asset lifecycle management (MIMOSA, 2022). It has published the Open Systems Architecture for Enterprise Application Integration (OSA-EAI™) specification compliant with ISO 13374 information architecture requirements and the Open Systems Architecture for Condition Based Maintenance (OSA-CBM™) specification compliant with condition monitoring and diagnostic processing architecture requirements. The current OSA-EAI release specifies the Common Collaborative Object Model (CCOM) for condition monitoring information and is defined as unbiased toward any single application and independent from physical storage or access methods. Also, past OSA-EAI releases defined the Common Relational Information Schema (CRIS) for OSA-EAI relational data store.

The information content is mentioned useful to operators, maintenance engineers, reliability analysts and management persons. Mathew et al., 2006, analyses MIMOSA OSA-EAI CRIS data model in detail from a system development and data modelling perspectives for a condition monitoring system implementation. They see integral issues in insufficient documentation on the specification and in database design choices that must be followed to retain a consistent data model. In addition, they present simplifications and additions to the data model such as removing unwanted tables and columns or linking new information for asset symptoms according to desired functions of the system under development. Also, benefits of CRIS data model normalization or denormalization are discussed.

2.4 Ontologies for information modelling

For information modelling in the cemented waste management, attractive ontologies are introduced in this section. Existing ontologies have been defined for a variety of different purposes. They can improve the utilization, management, and analysis of health monitoring data. Furthermore, by combining existing ontologies (ontology merging), a suitable composition can be formed for special requirements.

Ontologies are based on open-world-assumption (OWA). So, their nature is rather descriptive than restrictive in the information modelling. For the novel condition monitoring in the cemented waste management, ontologies can define a common terminology and demonstrate its use. Devices and different health monitoring data sources may use different terms and concepts. In this case, uniform concepts and terminology can be defined to facilitate data understanding and comparison. In addition, an ontology can be used to create a unifying framework that helps integrate different data sources. In this case, a more comprehensive picture of the state of the system is obtained. Ontologies help also to present the provenance of condition monitoring data, i.e., its origin and history. This is useful when investigating failure situations or performance degradation and their causes. In addition, ontologies enable semantic search capability. This makes it easier to make more

complex queries and search for more accurate data. Ontologies facilitate information sharing between different systems. With their uniform structure, data exchange and interoperability with different condition monitoring systems are easier to implement. In summary, ontologies may make it easier to implement a base for automatic decision-making and intelligent systems.

2.4.1.1 Provenance ontology (PROV-O)

PROV-O (PROV-O: The PROV Ontology, 2013) is a specification developed by the World Wide Web Consortium (W3C) that defines different concepts and the connections between them for presenting and sharing provenance information in a standardized way. Provenance information is information about the origin, management, ownership or history of a certain entity. The key concepts of the PROV-O ontology are Entities, Activities, Agents and their relations. Individuals and properties describe how entities are created, used, and how different operations and agents affect them over time. All in all, the PROV-O ontology is useful in information management and information systems where monitoring and understanding the history of data and its processing are important. Main classes and relationships are described in the Figure 4.

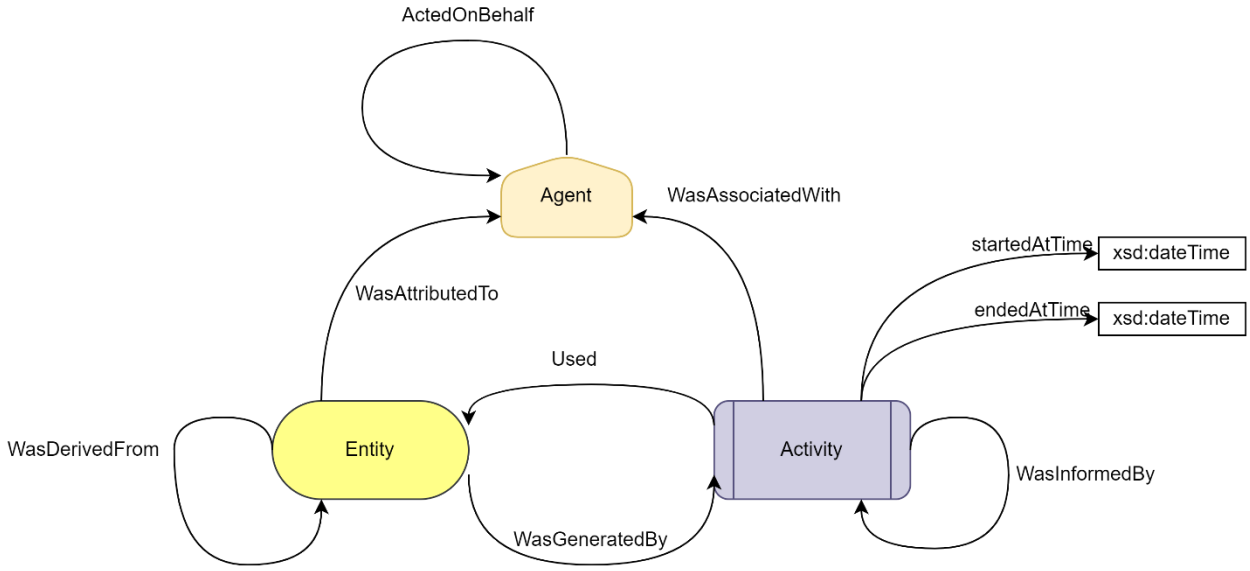


Figure 4. Classes of PROV-O.

2.4.1.2 PROV-O in condition monitoring

Here we describe the application of PROV-O in condition monitoring. Provenance information means when the data was created, which sensor or device collected the data, how the data was transformed, how the data was processed. PROV-O's Entity class refers to the object's components and sensor measurements, from which a unified description of the entities of the condition monitoring system is compiled. PROV-O's Activities describe operations that have an impact on the object's state or health monitoring data. Such are, for example, maintenance operations, cleanings, calibrations, which can be done by an agent. Agents are typically maintenance personnel or automation systems. The relations between individuals of Entities, Activities and Agents describe which function produced an individual of Entity, and by which Agent individual. PROV-O also enables data versioning and its derivation in situations where a data set has been derived from another data set with the help of some data processing. In addition, it enables environmental information to be described. Environmental information means the environmental conditions, such as temperature and humidity, that prevail in the storage.

Here we describe a brief example of a possible application of the PROV-O ontology in the context of the PREDIS project. Figure 5 describes the classes and the relationships between them. Here, the class symbols are the same as in Figure 4. Entities include MeasurementData, Feature, State, HealthIndex, Prognosis, and Recommendations. Activities include DataAcquisition, DataManipulation, StateDetection, HealthAssessment, PrognosticAssessment, and AdvisoryGeneration. Agents are a sensor and a digital twin.

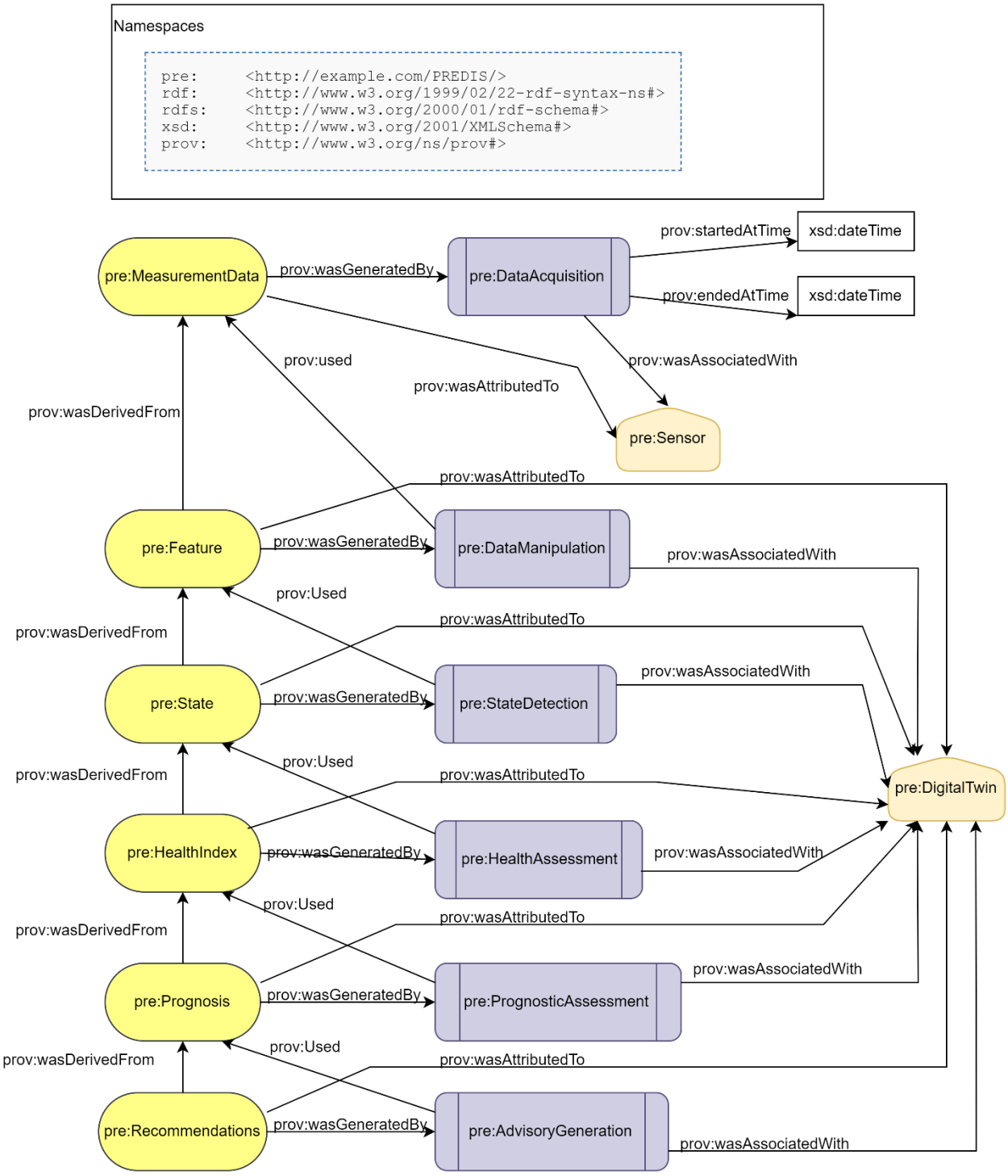


Figure 5. PROV-O example.

Reading Figure 5 from top to bottom, the Sensor Agent performs the DataAcquisition Activity, as a result of which the Entity MeasurementData is created. A time has been defined for this execution: startedAtTime and endedAtTime. In the next step, the DigitalTwin Agent performs the Activity

DataManipulation, which uses the MeasurementData entity. A Feature Entity is created as a result of the DataManipulation activity.

A basically similar structure is repeated downwards in the picture. The Agent executes an Activity that uses the Entity from the previous step. A start time and an end time have been defined for each Activity, which, however, have been left out of the picture for clarity. As a result of the activity, a new Entity is created. Relationships arise between entities created in this way, where the latter Entity is derived from the former. Thus, the provenance information and the entire history of the Entities will be described.

2.4.2 Ontology of Unit of Measure (OM)

OM (OM – Ontology of units of Measure, 2017) is an ontology that provides concepts and relationships between them for describing quantitative scientific information (e.g., units, amounts, dimensions). The purpose of this ontology is to facilitate consistent and interoperable description of quantitative data in different applications. The following concepts are central to OM:

- Unit
- Quantity
- System of Units
- Measure
- Scale
- Point
- Dimension

Here we describe a brief example of the use of the OM ontology in the context of the PREDIS project. Condition monitoring involves many different measurements of physical properties and phenomena, the exact definition of which is important. The example is narrow, as it contains only one measurement unit and its related classes and their instances. The ontology is described in Figure 6. Classes are depicted with rounded rectangles and their instances are depicted with sharp rectangles. Colour coding describes namespaces: OM is orange, PREDIS namespace pre is blue.

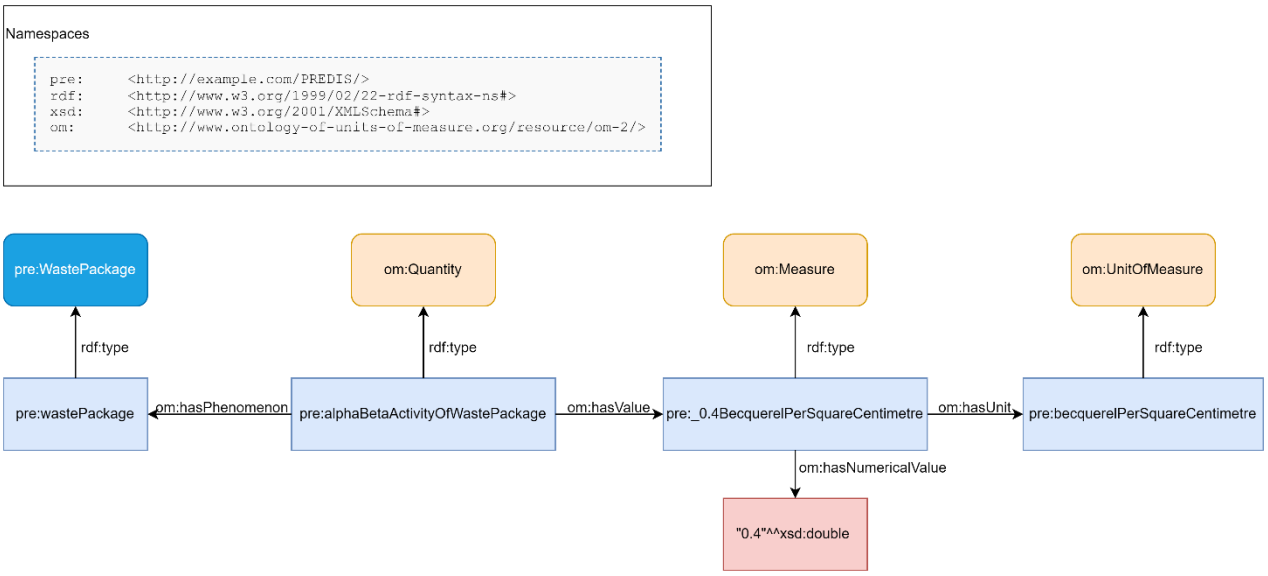


Figure 6. OM example.

The WastePackage class and its instance are on the left. WastePackage is associated with the phenomenon of surface contamination, which we want to measure. AlfaBetaActivityOfWastePackage is an instance of the rather broad class Quantity. The instance has

a measured value (hasValue relation), which is an instance of the Measure class. The instance has a numerical value and its unit of measure. The numerical value is 0.4. The unit of measure is defined in the UnitOfMeasure class of OM.

2.4.3 Semantic Sensor Network (SSN)

The Semantic Sensor Network (SSN) ontology is an ontology developed by the World Wide Web Consortium (W3C) that describes sensors, observations and related concepts. The SSN ontology is to improve interoperability and facilitate the integration of sensor data in different applications.

In SSN and SOSA, the most central concepts are listed (Semantic Sensor Network Ontology, 2017):

- Sensor: “Device, agent (including humans), or software (simulation) involved in, or implementing, a Procedure.”
- Observation: “Act of carrying out a Procedure to estimate or calculate a value of a property of a FeatureOfInterest”
- Feature of Interest: “The thing whose property is being estimated or calculated in the course of an Observation to arrive at a Result”
- Procedure: “A workflow, protocol, plan, algorithm, or computational method specifying how to make an Observation, create a Sample, or make a change to the state of the world (via an Actuator)”
- Sample: “Feature which is intended to be representative of a FeatureOfInterest on which Observations may be made”
- Sampling: “An act of Sampling carries out a (Sampling) Procedure to create or transform one or more”
- Platform: “is an entity that hosts other entities, particularly Sensors, Actuators, Samplers, and other Platforms”
- Property: “A quality of an entity”
- Actuator “A device that is used by, or implements, an (Actuation) Procedure that changes the state of the world”
- Actuation “An Actuation carries out an (Actuation) Procedure to change the state of the world using an Actuator”

A sensor refers to a device or system used to measure phenomena in the physical world. Observation refers to a process that measures a phenomenon or characteristic with the help of a sensor. Feature of Interest is the thing whose properties that the sensor is supposed to measure. Observation Procedure refers to the method or procedure required to make an Observation. Platform refers to the object to which the Sensor is connected. Property describes the measured property of the Feature of Interest.

The SSN ontology is a standardized way of presenting and linking information related to sensors, which facilitates the sharing and integration of data from different sensors and platforms.

Here we describe a brief example of the use of the SSN and SOSA ontologies in the context of the PREDIS project. The example is narrow and covers only a small part of the potential of the SSN and SOSA ontologies. The ontology is described in Figure 7. Classes are again depicted with rounded rectangles and instances with sharp rectangles. The namespaces are separated by colour coding.

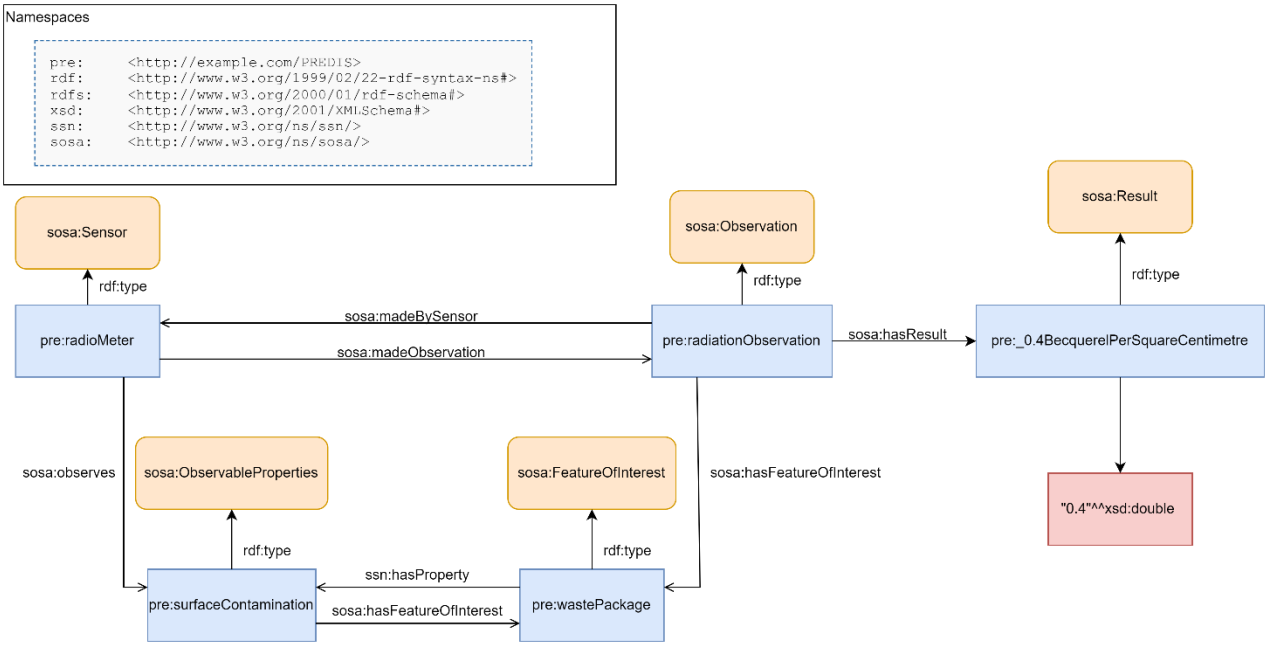


Figure 7. SSN and SOSA example.

The FeatureOfInterest class and its instance wastePackage are at the bottom center of the image. So the WastePackage instance is the thing whose measurable properties we are interested in. The instance has a relation to surfaceContamination, which is an instance of the ObservableProperties class. These describe exactly the phenomenon being measured. The Sensor class and its instance radioMeter are on the left. The radiometer has a observes relation with the observed phenomenon, i.e., surfaceContamination. The sensor makes observations. This activity is described by the class Observation and its instance radiationObservation. An observation has some result that contains a numerical value and its unit of measurement. These are described by SOSA's Result class and its instance.

2.4.4 Combining PROV-O, OM, and SSN

By combining different ontologies, a more comprehensive representation of the terms and concepts of the target area can be compiled. Combining the ontologies of Units of Measure (OM), PROV-O (Provenance Ontology), and Semantic Sensor Network (SSN) provides a more comprehensive representation of condition monitoring data, including units of measure, provenance information, and sensor-related information.

Here we describe a simple example of how the aforementioned ontologies (i.e., PROV-O, OM, and SSN) can be combined in the context of the PREDIS project. Classes are again depicted with rounded rectangles and instances with sharp-edged rectangles. The example is closely based on the OM and SSN ontology examples mentioned above. The PROV-O example is now formatted differently than above. The example ontology is described in Figure 8.

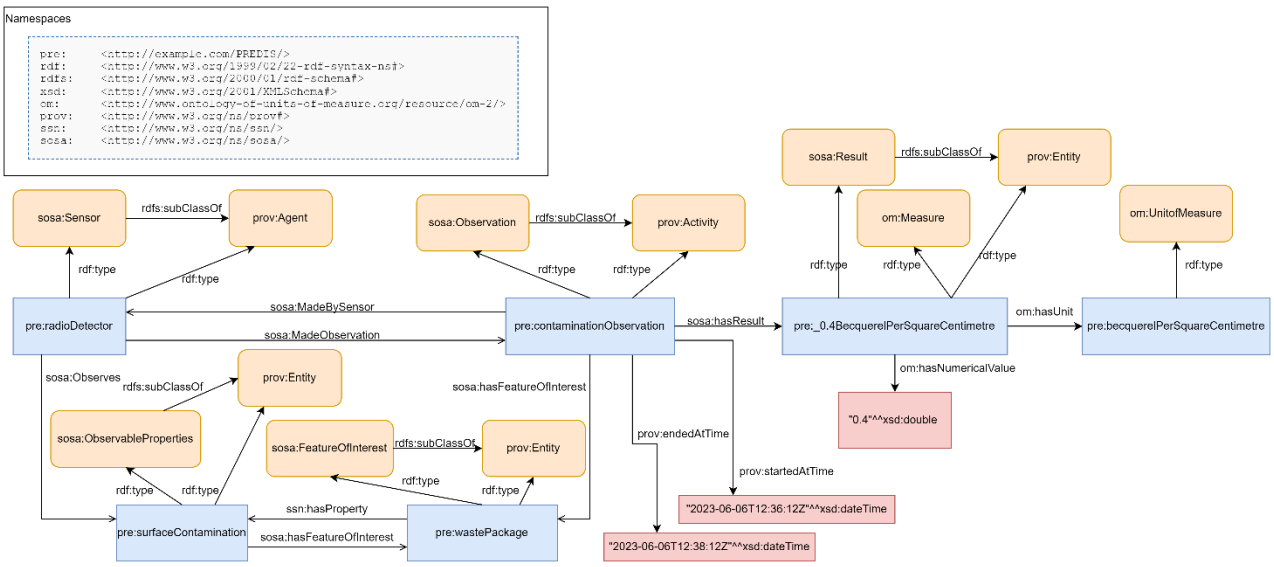


Figure 8. Example of ontology merging.

The ontologies SSN and PROV-O are aligned in such a way that many classes of the SOSA and SSN ontologies are subclasses of the PROV-O ontology. These relationships are modelled using the isSubClass relationship. For example, the Sensor class of the SOSA ontology is a subclass of the Agent class of the PROV-O ontology. In this way, the radiometer is an instance of both the SOSA Sensor class and the PROV-O Agent class. Thus, the relations of these classes can be applied to the instances. Otherwise, the interpretation of the diagram takes place in the same way as in the examples mentioned above, where individual ontologies were described.

3 Data management framework

3.1 High-level system description

System life cycle processes in systems and software engineering (ISO/IEC/IEEE 15288, 2015) serve as our process reference model and provide a process framework to identify processes under study. The PREDIS project identifies that measurement, system analysis, information management and decision-making processes are the most relevant ones (Figure 9). Figure 9 also maps processes (black-coloured text next to orange-coloured boxes) and systems (orange-coloured boxes) together and identifies the main information flows between the systems (grey-coloured boxes). The flow of information starts from the waste package (where the interesting monitored physical phenomena are happening), which the sensing and monitoring system tries to measure and, on the other hand, the digital twin tries to model for predictions. Both datasets are gathered in the data platform, which acts as the central repository for processing, storing, and transferring data between other enabler systems. The decision framework provides the measured and predicted information visually to the end user, helping them in the decision management process.

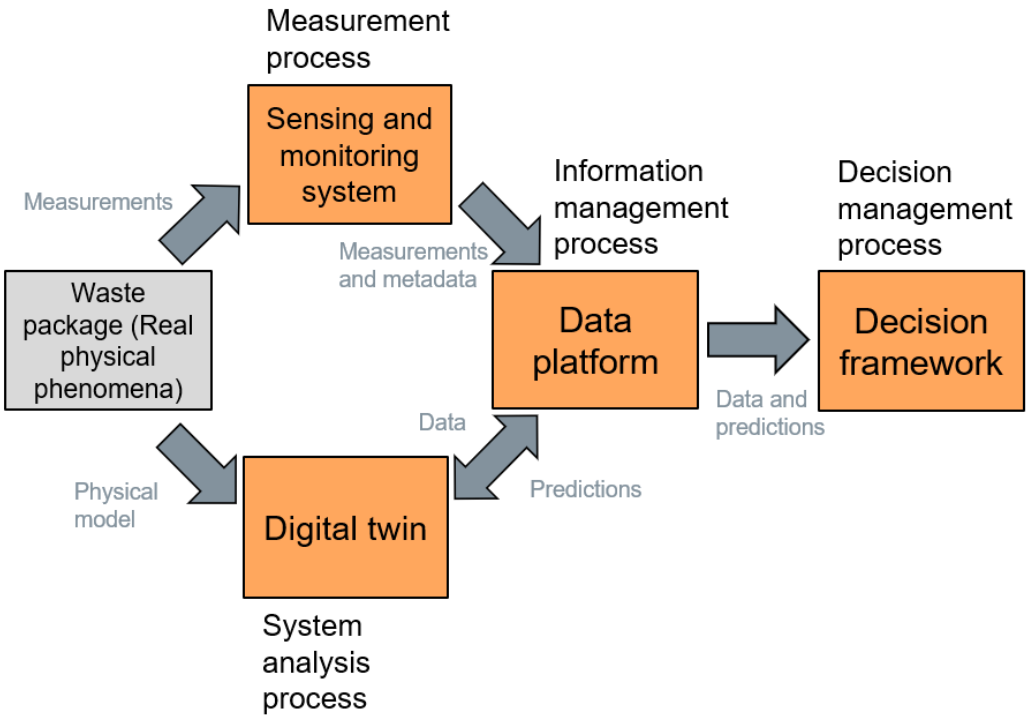


Figure 9. Data management framework high-level system architecture.

On a high-level, the generic condition monitoring system functionality blocks are described in Chapter 2.3. Here they are allocated into the four data management framework subsystems, which are interacting with each other as shown in Figure 9. The four subsystems are briefly described here, before they are further described in the following chapters.

- **Sensing and monitoring system** handing the measurement process: the development of this subsystem is generally out of the scope of this task; it has been the focus of Task 7.3 within PREDIS project. However, there has been close communication between the partners of these tasks, as the measurement are the input for the data management platform of Task 7.5. All latter systems have been fitted to manage the data coming from the measurement process. There exists PREDIS project reports and deliverables describing this subsystem in more detail. This subsystem is fulfilling the functionalities of the *data acquisition block* and the *data manipulation block* from Chapter 2.3

- **Data platform** handling the information management process: this subsystem is the central data repository of the data management framework, interfacing with the various data input and output flows as shown in the Figure 9. This subsystem is further explained in Chapter 3.2. This subsystem is fulfilling the functionalities of the *state detection block* and the *data manipulation block* from Chapter 2.3.

- **Digital twin** handling the system analysis process: the development of this subsystem is generally out of the scope of this task; it has been the focus of Task 7.4 within PREDIS project. It aims at condition prediction and the description of the evolution of the waste form. However, the linking of these two tasks has been discussed during the project, and general principles how these tasks are supporting each other are clear. There exists PREDIS project reports and deliverables describing this subsystem in more detail. This subsystem is fulfilling the functionalities of the *prognostic assessment block* and the *health assessment block* from Chapter 2.3.

- **Decision framework** handling the decision management process: this subsystem is the user interface to the data management framework and aims to assist the decisions makers in various ways with information about the condition of the monitored packages and complementing that information with the prediction capabilities of the digital twin subsystem.

This part is further described in Chapter 3.4. This subsystem is fulfilling the functionalities of the *advisory generation block* and the *health assessment block* from Chapter 2.3.

Early in the project a decision was made to use Microsoft Azure, a popular cloud computing platform, as the common platform for concepting and developing the system prototype. A platform with access from all project partners was needed, and Azure offers several key features and advantages, for example: scalability and flexibility, IoT integration, security and compliance, global reach and availability, and good developer tools and ecosystem. As a main developer environment, a virtual machine within the Azure platform was established, including the needed database selections done in Chapter 3.2, also the Azure data lake was used to store data.

The following chapters further describe each of the system elements in the focus of Task 7.5.

3.2 Data platform

This chapter reports the work done in PREDIS Subtask 7.5.1 by partner ANN, which focused on the development of the data platform part of the data management framework.

3.2.1 Design and functionalities

The activity to be performed can be organized into three main tasks with different objectives:

1. Develop a platform that will foster the seamless connection between devices and sensors from different manufacturers, including those used in the monitoring system.

2. Implement a configurable Human Machine Interface (HMI) into the platform, which provides the real-time status of field sensors, time trends, alarm thresholds, etc., and is safely accessible to third parties (i.e., safety authorities, fire brigades, etc.)

3. Develop a “dashboard concept” for the platform that enables model prediction of event evolution (i.e., concrete waste package evolution and/or deposit) to be assessed alongside real-time sensor measurements

4. Develop a reference database for the platform containing waste-condition information from various stages throughout the waste lifecycle, starting at the acceptance of the (often unconditioned) waste and including the impact of the conditioning (i.e., cementation) and the evolution of the waste during pre-disposal storage

The design activities have been involving different organizations as contributors.

3.2.2 Implementation and technology selection

As in any project, the data-management project requires the collection of input data and requirements, in particular:

a. Definition of data to be managed as input and output

b. Definition of data life management requirements

c. Database availability and evolving technologies

d. API Specification.

Based on above, case studies can be set-up to highlight advantages and disadvantages of any potential design choice.

3.2.3 Definition of Input Data

Preliminary definition of possible input data is required to assess properly the data management process.

In Subtask 7.5.1, it can be assessed that the essential input data set to manage is:

A. Organization which is originating the data to be managed (PREDIS, other EU projects, End User).
B. Format of data to be managed («raw», «formatted» and metadata), in particular:
    1. **«raw»=** typical images and/or large quantities of data acquired in a short time
    2. **«formatted»=** data acquired from continuous measures to be transferred in a declared format
    3. **metadata =** data almost immutable in the time characterizing:
        1. The waste package from its generation to its final allocation
        2. The sensors and/or associated Data Acquisition Units (DAU)
        3. The correlations between the sensor DAU and the waste package
        4. The change of working conditions of measuring device.
C. Data Flow Criteria:
    1. Hardware to be interfaced for data acquisition should not represent a specific constrain since it is always possible to adopt a dedicated gateway
    2. Frequency of data acquisition can range from 100 Hz for few seconds to once a day/week/months for long time.
    3. Data properties have to be fixed as formats so that it is not required to edit the file in the same application that was used to generate it. In this class are included: structured CSV, JSON, PNG, JPG, TIFF, PDF
    4. Data shall be acquired and archived so that queries are user-friendly.

3.2.3.1 Metadata Definition

A reference nomenclature for the metadata has to be defined in agreement also with the schemes presented in other WPs (i.e., 7.3, see https://github.com/predis-h2020/metadata)

The current proposal is to define four main “Classes”:

1. The class “**WastePackage**”
2. The class “**Sensors**”
3. The class “**Images**”
4. The class “**Nodes**”

Each class can be sub-divided in different sub-classes.

The “**Waste Package**” Class is formed by three second level classes, by considering also End user indications (see ref (7)):

“Basic Unit” subclass:

1. ID
2. Waste Type (VLLW, LLW, ILW, MLW, HLW)
3. Waste Original Inventory
4. Waste Treatment (i.e., grouting)
5. Shape (i.e., Drum, Barrel)
6. Dimensions (Inner, Outer)
7. Volume
8. Concrete Filling Grade (only if grouted)
9. Alpha/Beta Surface Contamination (Bq/cm^2)
10. External Dose Rate (mSv/h)

11. Waste Inventory (vector ID)
12. Total Gamma Activity (Bq)
13. Gamma spectra (link)
14. Place of Origin
15. Date of production
16. Date of characterization
17. Arrival Date
18. Loading Date
19. Picture Link
20. Drawing Link
21. Presence of sensors inside (ID list)
22. Presence of sensors outside permanent attached (ID list)
23. Presence of DAU (list)
24. Coordinates of sensors inside (if any)
25. Coordinates of sensors outside (if any)
26. Coordinates of DAU (if any)

"Pallet" subclass: a set of Drums/Barrels placed on a pallet (1, 2, 4 or 6 drums)

1. Pallet ID
2. Pallet Size
3. Basic Units IDs
4. Place of final assessment
5. Presence of DAU (list)
6. Coordinates of DAU (if any)

"Waste Package" Subclass:

7. Type: a Basic Unit or a Pallet
8. Storage Tag
9. Storage coordinates (cartesian x,y,z or vault coordinates)
10. Movement history tracking (site list)
11. Date of loading in the vault
12. Date of last check
13. Date of planned next check
14. Array: set of waste packages in a specified direction (usually z)

The "**Sensors**" Class is characterized by one subclass:

"Sensor" subclass:

1. Sensor ID
2. Sensor Type
3. Position (inside or outside basic unit, environmental)
4. Last calibration date
5. Calibration certificate number
6. Calibration factor
7. Product name /sensor manufacturer
8. Total number of measurement type (maximum 2: P&T, RH&T, P&RH, Gamma&Neutron)
9. For each measure: measurement type, unit of measurement value, min –max value
10. Spatial location (referred to the basic unit or to the Pallet or Vault)
11. Origin Vector (default Z axis parallel to drum axis, origin on the drum bottom)

The "**Nodes**" Class is characterized by two subclasses:

"Sensor Node" (or Data Acquisition Unit) subclass

1. Product ID
2. Product Type
3. Manufacturer
4. HW Serial Number
5. SW Firmware version
6. Location
7. Part Number
8. Last Inspection date
9. Connected Sensors List
10. Total channels available (maximum 12)
11. For each channel
    1. Channel ID
    2. Connected Sensor ID
    3. Associated Measures
    4. Measurement Unit
12. Working status ("available"/"out of service")

"Gateway" subclass.

1. Product ID
2. Serial Number
3. MAC address
4. Connected "Sensor Nodes/DAU" list
5. Working status (available /"out of service")
6. Communication protocol

The class "**Images**" is characterized by the hyperlink to any information to be stored as raw data in graphical form by including pictures, drawings, calibration certificates, radwaste characterization reports, etc.

3.2.3.2 Default data format and access options

The formatted data can be considered sources for "Time Series" data acquisition and handling. The current main data to manage in this project are provided from Task 7.3.

- BAM:
    - Concrete embedded RFID Sensors: RFID sensor placed inside the waste drum to monitor moisture and temperature.
- UNIPI/CAEN:
    - External Dose Rate Sensors: gamma & neutron plus RFID techniques
- INFN:
    - Gamma dose rate field associated to each drum monitored by using optical fibres put outside the drum

The data flow is coming at a fixed frequency with an associated timestamp.

3.2.3.3 Dummy data options

Data management in management nuclear radwaste (see Data management in https://doi.org/10.1016/j.pnucene.2022.104251) requires to enable consistent accessibility, delivery, governance, and security of data to meet an organization's requirements using tools including master data management, data virtualization, data catalogue, and self-service data preparation and wrangling.

A data management program implementation should be set-up using the following characteristics: Established data governance controls that provide security by limiting access to data only to authorized users, making it easy to identify the data you're looking for with clear metadata. Easily accessible data, including streaming, transactional, structured, and unstructured data. An infrastructure that can evolve as business needs change. The ability to work with existing and legacy technologies without having to go through the expensive task of "ripping and replacing." Consistent and controlled data sharing across business domains, allowing for data use in operations, analytics, and governance. Data quality that measures up in these five key areas:

1. Validity: The data conforms to the syntax (range, format, type) of its definition.
2. Consistency: When comparing two or more representations of an object or event, there are no differences.
3. Uniqueness: No copied data records.
4. Accuracy: The data can correctly describe the "real-world" object or event in question. Completeness: All relevant data is included.
5. Timeliness: The data is up to date and represents reality from a very recent point in time.

The data management program can be addressed at different technologies:

1. Build Data Warehouse Structure
2. Use Analytical Database
3. Automate with Orchestration Tools
4. Monitor System with Alert Notifications

3.2.3.4 Communication sensor testing parameters

Recent advancements in wireless communications and microelectronics, in addition to subsequent price reductions, have enabled the utilization of Wireless Sensor Networks (WSNs). The selection of a particular communication technology for radiological waste package monitoring applications strongly depends on the characteristics of the IoT environment, the number of sensor devices, energy requirements, network scalability, and so forth.

Some reference data has to be transferred during the real time communication with the real time database to be able to check if the data received are corrected and transmitted safely, in particular:

a) the MAC address of the device sending the data
b) the strength of the signal associated to the device where the sensor node is transmitting to the associated router.
c) a unique authorization key to write in a file.

There is also a set of characteristics information associated to each sensor node, in particular:

a) the data-rate (kbps)
b) the sampling rate (Hz)
c) the total number of sensors communicating with it
d) the total number of associated ADC (none, 12-bit, 16-bit, 24-bit)
e) the power consumption (W)
f) the level of “privacy”
g) the maximum latency allowed (ms).

3.2.3.5 RAW data

The RAW data are essentially imaging data. Measures originating from an image are to be considered periodic measures which need to be validated on the screen:

- in open format (JPG, .PDF, .TIFF, BMP...)

- in proprietary format (they cannot be open without the program used to generate it)

All imaging data produced can be made available from the originating organization by uploading to a common storage (i.e., uploaded on VTT MS Azure BLOB (working as an archive) and a link to the space storage (i.e., MS Azure BLOB) could be associated to the Waste Package meta description.

The RAW data in PREDIS are mainly expected to be produced by BAM in Task 7.3. They are coming from the following measurements:

- High frequency ultrasonic echo: Data are produced as .lbv file which contains binary data storage of all original time-domain signals and the corresponding configuration.
- Low frequency ultrasonic echo to inspect the container concrete filling in term of structure, presence of voids, objects, and cracks. Sensors are placed outside the container. Data management issues are like high frequency ultrasonic echo. Information needs to be validated visually.
- Ultrasonic Monitoring (place sensors inside the package filled with concrete to monitor changes, crack development, stiffness).
- Digital Image Correlation (DIC) to verify container surface inspection from remote to check structure, cracks, deformation.
- Acoustic Emission analysis (AE) to inspect a container by using external sensors to detect changes in concrete during filling and to observe crack development. Approaches in recording and analysing AE signals can be divided into two main groups: parameter-based (classical) and signal-based (quantitative) AE techniques.

RAW Input data can be originated also from other organizations and/or in the frame of other EU projects, in particular:

a. UJV (Organization):
    - Gamma Scanner: This technology is addressed to quantify gamma emitters (Cs, Co) from remote for inspection purpose inside a container.
    - Radiography: The based concept is to use a 9 MeV LINAC in order to check the density distribution
b. MICADO Project (https://www.micado-project.eu/)
    - Gamma camera (Orano DS, ENEA) addressed to hot spot search from remote for inspection purpose.
    - Gamma spectrometry (Orano DS, ENEA) to quantify gamma emitters (Cs, Co) from remote for inspection purpose.
    - Neutron active measurement (Orano DS, CEA) to quantify nuclear materials (Pu, U) in non-cemented package remotely.
    - Neutron passive measurement (Orano DS, CEA) to quantify nuclear materials (Pu, U) in the non-cemented package remotely.
    - Photofission measurements (Orano DS, CEA) to quantify fissile materials in the cemented waste package from remote for inspection purposes. Fissile materials should not disappear once the package has been realized.
    - SciFi technology (Orano DS, INFN) Gamma dose rate field associated with each drum is monitored by using optical fibers put outside the drum
    - SiLiF technology (Orano DS, INFN) Neutron dose rate associated with each package is monitored by using SiLiF detectors put outside the drum
c. CHANCE project (https://www.chance-h2020.eu/) to understand current characterization methods and quality control schemes: Calorimetry as an innovative non-destructive technique to reduce uncertainties on the inventory of radionuclides; Muon Tomography to address the specific issue of non-destructive control of the content of large volume nuclear

waste; Cavity Ring-Down Spectroscopy (CRDS) to characterize outgassing of radioactive waste.

3.2.4 Definition of Output Data

Data storage in a database can be retrieved by a query, which is a request for data or information from a database table or combination of tables. This data may be generated as results returned by Structured Query Language (SQL) or as pictorials, graphs, or complex results, e.g., trend analyses.

3.2.4.1 Queries for Prediction

Queries for prediction are also called “data mining” and are influenced from different factors. The main is the table organization and if the selected data are present in column or not. For data mining the organization of data shall allow to select a “column” and a “field”.

3.2.4.2 Queries for Dashboarding

To create a real-time SQL dashboard, an AI tool that can fetch live data from relational databases is needed. Alternatively, query results from SQL can be exported as a CSV and plugged into Google Data Studio or Excel, but this is not real-time.

In general, the best way to proceed is to use a third partner software package; in particular, if people are using MS Azure the best way to proceed is to use MS Data Explorer (see https://db-engines.com/en/system/Microsoft+Azure+Data+Explorer) or if people are using influxDB (www.influxdb.com) the best way could be to write a script by using flux syntax.

3.2.5 Definition of Life Management Requirements

Data lifecycle management (DLM) (see https://www.ibm.com/topics/data-lifecycle-management) is a policy-based approach to manage the flow of an information system's data throughout its lifecycle: from creation and initial storage to when it becomes obsolete and is deleted.

DLM products automate lifecycle management processes. They typically organize data into separate tiers according to specified policies. They also automate data migration from one tier to another based on those criteria. As a rule, newer data and data that must be accessed more frequently is stored on faster and more expensive storage media, while less critical data is stored on cheaper, slower media.

DLM can be broken into multiple phases that provide a framework for working with data throughout its lifecycle. Although different resources identify these phases in various ways, they often follow a structure like this:

- Generate and collect data. Structured and unstructured data is continuously being created by users, devices, applications, machinery, IoT devices and other means. The way in which that data is captured depends on how it is generated and the types of data and applications. In some cases, not all generated data is collected. For example, machinery data might generate enormous amounts of sensor data, but only anomalous data is collected.
- Store and manage data. Data must be stored in a stable environment and properly maintained to ensure its integrity, security, and protection. During this phase, the data is typically processed in some way, such as being encrypted, compressed, cleansed, or transformed. This phase also makes sure that systems are in place to ensure availability and reliability and to implement redundancy and disaster recovery.
- Use and share data. Data is valuable only if authorized users can work with it as needed to carry out their day-to-day operations. During this phase, users access and modify data as needed and carry out other data-related operations, such as collaboration, business

intelligence, advanced analytics, or visualization. Data usage can also result in additional data being created, which must then be stored and perhaps further processed. In effect, this phase is what enables authorized users to be able to do their jobs.

- Archive data. At some point, data is no longer needed to support the everyday applications and workflows, in which case, the data can be archived in a secure, long-term storage system such as tape storage or a cloud platform. The data might still be needed at some point for compliance, analysis, reporting, or other purposes, which means it must remain available and viable, but it isn't required for daily operations. The data should also be fully protected, just like active data.
- Destroy data. When data has reached its end-of-life, it can be permanently deleted, but this must be done securely and without violating applicable data protection regulations.

The third stage might result in additional data being generated. The first three stages often occur simultaneously, with data being continuously generated, collected, stored, managed, and made available for authorized usage.

3.2.6 Databases Trade-Off Analysis

More than one type of database is expected to be necessary with different functions and/or performances, in particular:

a) Metadata are more addressed to SQL database (structured mode)
b) Times series or “formatted” data are more addressed to “not only SQL” databases (unstructured)

The reason of this choice will be explained in the next paragraphs.

Databases can save information in structured mode or unstructured mode:

- Structured: MS SQL, MySQL, MariaDB, PostgreSQL, SQL Lite (Engine), etc...
- Unstructured (not only SQL): MongoDB, Amazon DynamoDB, SQL Lite 3 (Engine), etc...

3.2.6.1 SQL Databases

SQL stands for Structured Query Language. Developed in the 1970s, SQL is the industry-standard language for organizing, editing, and managing relational databases.

SQL databases are table-based. This means they organize and store data in tables with predefined categories or columns. Relational databases contain structured data, such as names, email addresses, and phone numbers. A relational database matches data by using common characteristics found in the dataset, resulting in a group called a schema. Using SQL, you can add, delete, search, update, and organize data records in a relational database. It has since become an industry standard, with many popular relational databases using SQL, including MySQL Database, Oracle, and Microsoft SQL Server.

SQL databases are all pretty similar, but each one uses a slightly different version of the SQL language. Here are some common SQL databases with a few details about them:

- MySQL https://www.mysql.com/
  - Free and open-source
  - Available for all major platforms
  - Huge community of developers
- Oracle https://www.oracle.com/
  - Commercial
  - Also has a procedural language
  - Advanced transaction control

- PostgreSQL https://www.postgresql.org/
  - Free and open-source
  - High ACID compliance
  - Has JSON fields for unstructured data
- Microsoft SQL Server https://www.microsoft.com/it-it/sql-server
  - Commercial database system by Microsoft
  - Also has a procedural language for writing procedures
  - Works on Windows and Linux
- SQL Lite 3 https://www.sqlite.org/index.html
  - Free and open-source
  - Serverless database management system that can be embedded directly into an application
  - Not suitable for multiple user access (i.e. networking with different IP)
  - Limited database size (i.e. each table < 2 GB)
- MariaDB https://mariadb.org/
  - MySQL code modifications made in to improve its performance, scalability, and features
  - GPL 3.0 license allows easily integration into closed source software applications
  - Support for stored procedures, triggers, and views (as Microsoft SQL)
  - Community driven

Since each SQL database has different characteristics, the choice is depending from the project. In general, open-source database should be preferred.

3.2.6.2 NoSQL Databases

NoSQL stands for Not Only SQL, or Non-SQL, and refers to non-relational databases. While SQL itself is a query language that communicates with databases, NoSQL is an adjective used to describe a non-relational database that doesn't require the SQL language.

In a NoSQL database, unstructured data can be stored across multiple servers and processing nodes. Because non-relational databases don't require fixed table schemas, they can not only store structured data but also semi-structured and unstructured data. This makes them easier to scale and manage, especially for organizations with massive data storage needs.

Some examples of NoSQL databases include MongoDB, RavenDB, Cassandra, BigTable, and CouchDB. SQL Lite 3 is declaring itself as a NoSQL database.

Most NoSQL databases can contain the following four data types:

- Document-oriented stores. These let a key pair with a document. Documents can hold a variety of objects like key-array and key-value pairs, as well as other documents.
- Key-value stores. These are simple databases that store information in the form of attributes (keys) and values. In some cases, values can have types like "string" or "integer."
- Graph stores. These store data about networks in a graph-oriented format.
- Wide-column stores. These help with handling large amounts of data in the form of columns.

With auto-sharing, NoSQL databases can share data across servers without complex programming or code. This balances the processing load of storing and managing data across multiple servers, whereas SQL databases rely on a single server. This also makes NoSQL databases more secure in the event of a server crash — if one server goes down, the others in the system will still function and be able to access and store data.

Modern programming is often iterative, meaning that programmers gradually add to a database or application over time. An iterative programming approach can be a challenge when using SQL

servers because of their fixed schema structure. This makes it time-consuming to add to or change the structure of the data and records in a SQL database.

NoSQL databases are more flexible and compatible with iterative programming. Because defining schemas isn't necessary at the start and NoSQL databases can handle more than one type of data, programmers can add or change the structure of data and records as they go. These minor changes don't disrupt the entire system or require large data transfers.

All SQL databases are pretty similar to each other. But the similarity between NoSQL databases often stops at not using SQL. There are quite a few types of NoSQL databases, with MongoDB being the most common. Here are some common NoSQL databases:

MongoDB (https://www.mongodb.com)
- Stores data as JSON documents
- Can scale quickly
- Is really fast for simple queries

Redis (https://redis.io/)
- Stores key and value sets
- Is very fast
- Is commonly used for high-speed caching

DynamoDB (https://aws.amazon.com/it/dynamodb)
- Provided by Amazon Web Services
- Similar to MongoDB
- Has a simple JavaScript Interface

CouchDB (https://couchdb.apache.org/)
- Data stored as JSON documents
- Can be queried from a web browser
- Built-in conflict resolution.

3.2.6.3 Blockchain as tracking system

Blockchain technology enables the tracking and tracing of waste from source to disposal site, creating a transparent and tamper-proof record of the waste's journey. This enhances accountability, deters fraud, and ensures compliance with environmental regulations.

By attaching a digital "tag" (i.e., "token") to each waste item or batch, its journey can be tracked and recorded on a decentralized and immutable ledger. This means every participant in the waste management chain – from producers to transporters to disposers – is held accountable for their part in the process. Any attempts to tamper with the records would be immediately apparent, providing a robust deterrent against fraudulent activity.

When data can automatically identify and correct itself based on coded business logic (i.e. smart contracts) and consensus, participants are intrinsically able to trust it. When two businesses work together, they almost never share a single database with a single set of records, because the database is being maintained and updated by a database administrator (DBA). That DBA is being paid by one of the companies and thus has a stake in the success of one company but not necessarily the other. If they want to make a change that benefits their company, the other company will never know. Alternatively, on a more nefarious note, if a competitor decides to pay off the DBA, they can make any change they want to the database without either participant ever knowing.

When blockchain technology is incorporated into the data process, you remove the single point of failure, in this case the DBA, and ensure that if one of the participants makes a change it is immediately corrected by the other participants. After the data corrects itself, the unalterable record

of changes will also indicate which participant tried to make the change. With the data process secured, a business can not only trust the data shared between the companies they are working with but can even trust the data shared by competitors.

The MIT (Massachusetts Institute of Technology) gives the possibility to generate “smart contracts” for a single project in the frame of Django rest framework for the Internet of Things (IoT) based on Ethereum (see https://github.com/SandroLuck/SCGenerator) .

3.2.6.4 NoSQL vs SQL

There is an intense debate between developers regarding the merits of NoSQL and SQL system. The main difference between the modes to operate are here in reported.

Database type

- SQL databases are relational databases (RDBMS), meaning they store data in tables with predefined columns and rely on fixed table schemas to relate data records to each other.
- NoSQL databases are non-relational (or distributed) databases, meaning they do not require tables. Non-relational databases can store unstructured data using documents, key-values, graphs, or columns.

Fixed schema vs. dynamic schema

- SQL databases require a fixed (or predefined) schema. This means you must define the structure of your data first before working with it. All data must follow that same structure. This requires a lot of preparation upfront. Changing the structure (or schema) of your data is difficult and time-consuming, as it requires starting over with a new predefined schema and updating all your records to match the new schema. Even small changes may require system downtime or reduced service for a period while the database gets updated.
- NoSQL databases use dynamic schemas, meaning you can create data without having a defined structure in place. Each data record can have a unique structure, typically organized as a document, graph, column, or key-value. This gives you far more flexibility, as you don’t have to spend time upfront defining a schema, and you can change and update the schema as you go. Change management is far easier and rarely requires any system downtime for small adjustments in records and structure.

Scalability — vertical vs. horizontal scaling

- SQL databases scale vertically. This means that the database uses a single server. To scale (or increase the load) on that server, you can add RAM, add or upgrade the CPU, and add or upgrade the SSD. The only way to scale a SQL database for increased data is to improve the server’s capacity or purchase a larger, more expensive server that can handle more data storage.
- NoSQL databases are horizontally scalable. They rely on nodes that can share data and processing power. This means that you can add more servers to a NoSQL database to handle more data (increase the load). NoSQL databases can use multiple servers and share data across them. Because they use low-cost hardware, it’s often less expensive to scale NoSQL databases, and it requires little or no application downtime. This makes it easier to scale as the amount of data increases and means that NoSQL databases can become far larger than SQL databases, which are limited to only one server.

Data volume capacity

- While SQL databases can store millions of records, they do hit a storage limit eventually. One server can only hold so much data before it hits its maximum capacity, no matter how large the server. Purchasing increasingly larger servers also brings steep financial

commitment, meaning most companies will hit a maximum server size that they can afford to use for a relational database.
- On the other hand, NoSQL databases can handle a far higher volume of data, as they can spread data across thousands of connected servers. For organizations that need to store massive amounts of unstructured data, NoSQL databases are the obvious choice. They can handle far more data than SQL databases, with almost no impact on application performance.

Caching
- SQL databases need a separate infrastructure to cache data, which requires additional hardware and slows the performance of the databases.
- NoSQL databases can cache data directly within the system memory, resulting in much higher performance.

Advantages of NoSQL databases:

Scaling
- For large organizations, the relationships and tables that make up SQL databases can number in the millions. Combine this with millions of users performing lookups in these tables, and the system can suffer major performance issues. This is one of the main reasons organizations like Google and Amazon have switched to non-relational database systems.

Complex data management
- In addition, large-scale programming projects using complex data types and hierarchies, such as XML and JSON, are difficult to incorporate into SQL. These data types, which can contain objects, lists, and other data types themselves, do not map well to tables consisting of only rows and columns.
- Maintaining high-end relational database systems is expensive and is only possible with the assistance of highly skilled database administrators. This requires a significant investment for organizations.

Maintenance
- On the other hand, NoSQL databases require less management overall. Features like automatic repair, easier data distribution, and simpler data models mean that NoSQL databases require less administration and tuning than SQL databases.
- NoSQL databases typically use clusters of cheap commodity servers to manage increasing data and transaction volumes, while relational databases tend to rely on expensive proprietary servers and storage systems. So, in general, storing and processing data costs less in a NoSQL database than it does in a SQL database.

Here is a breakdown of when the use a NoSQL database is preferred:
- The traditional RDBMS model performance is not sufficient.
- A flexible data schema is needed
- Data from distributed sources need to be logged
- The constraints and validation logic, that a SQL database provides, are not needed.

Advantages of SQL databases

Support
- Customer support is currently stronger for relational databases because they have been around longer and have a global community of users. Relational database vendors also provide a higher level of enterprise support.
- Because NoSQL databases are newer, small start-up companies provide NoSQL system support. Although NoSQL is growing, the current providers lack the global reach, resources, or credibility of Oracle, Microsoft, or IBM — the big names associated with SQL.

Complex queries

- While NoSQL databases have evolved to meet the scaling demands of modern Web 2.0 applications, their query languages are harder to use for ad-hoc query and analysis.
- It is much easier to code a SQL query, which makes ad-hoc analysis a simple task. In NoSQL, even a simple query requires significant programming expertise. NoSQL has a steeper learning curve, especially for users accustomed to working with Excel and SQL.

User-friendliness

- Currently, the complexity of NoSQL makes it a good fit primarily for large technology companies that collect massive amounts of unstructured data and can afford to pay a trained database administrator with knowledge of NoSQL.
- Smaller projects and organizations typically function just fine with SQL. While it may be difficult in some cases, it's possible to map complex objects using tables. There are powerful tools available, such as the Oracle database system, which are very effective.

Here is a breakdown of when the use of a SQL database is preferred:

- A simple query language to interact with data is needed
- Use joins and complex queries is required
- Data is strictly structured, and that structure will not change often.
- Business rules depend on transactions.

3.2.6.5 Blockchain vs Centralized Databases

The primary difference between a blockchain and a database is centralization. While all records secured on a database are centralized, each participant on a blockchain has a secured copy of all records and all changes so each user can view the provenance of the data.

3.2.6.6 Databases Selection

The choice between SQL and NoSQL is related comes down to a few key factors.

The type of data used:
For structured data, SQL is likely the right choice. SQL databases work well for transaction-based needs, such as customer relationship management (CRM) solutions and accounting tools. A relational database will store this data in a structured way, with each row representing a distinct record and each column a distinct attribute in that data.

On the other hand, if primarily unstructured data that won't fit neatly into a table format or the nature of the data to be handled in not established, a NoSQL database would be a better option. SQL databases cannot handle unstructured data and are less flexible when changing the structure of data in the middle of a project.

The need to be compliance with:
If compliance with ACID (Atomicity, Consistency, Isolation, and Durability) is required, an SQL database is best. Because relational databases store records in a fixed table schema with distinct relationships between rows and columns, they are well suited to ACID compliance.

If compliance with the ACID model is not required, and you need more flexibility, a NoSQL database may be a better fit. NoSQL databases follow the BASE model (Basic Availability, Soft state, and Eventual consistency). This makes the database more flexible and scalable and allows the user to change his data structure mid-project with little or no downtime for its application.

How query on data will be performed:

How often the user needs to query his data should be considered. It's far easier to write queries with SQL than with a NoSQL query language. Querying a NoSQL database requires writing complex queries. If the user will need to query his data quickly or the user responsible for querying is not an advanced programmer, SQL is a better fit. If the user won't need to query his data often or can afford the skill set of a NoSQL data analyst, then a NoSQL database may be preferred.

What are user's future data needs:
It is important to consider how user organization expects to scale moving forward. If there is the need to handle exponentially larger volumes of data, a NoSQL database is easier and less expensive to scale up.

SQL databases are more costly to scale and often require system downtime during the data migration process.

Conclusion:
Based on the above considerations, it has been decided in the frame of Subtask 7.5.1 to choose as SQL database reference choices to investigate in case studies: Microsoft SQL, PostgreSQL, and SQL Lite 3 to associate to «metadata» management.

For "time series" and or formatted data, it has been decided to use a NoSQL database in an indirect form, i.e., use a commercial product with NoSQL databases inside and it offers the capability to facilitate the user in write AP named InfluxDB (www.influxdata.com).

It must be pointed out that Microsoft SQL is a typical commercial package while PostgreSQL and SQL Lite 3 are open-source packages and hence candidates to be more flexible to build API and to be interfaced with InfluxDB by the user itself.

The basic concept is the possibility of performing development on local workstations and then dockerizing to make it callable on the cloud.

3.2.7 API Specification

3.2.7.1 I/O string data format

Many formats can be chosen, but currently, the reference one is worldwide the JSON format, which is considered a valid alternative to XML. The JSON format is syntactically like the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.

Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.

- JavaScript has a built-in function for converting JSON strings into JavaScript objects: JSON.parse()
- JavaScript also has a built-in function for converting an object into a JSON string: JSON.stringify()
- When storing data, the data must be a certain format, and regardless of where it has been chosen to store it, text is always one of the legal formats.
- JSON makes it possible to store JavaScript objects as text.

JSON syntax is derived from JavaScript object notation syntax:
- Data is in name/value pairs
- Data is separated by commas

- Curly braces hold objects
- Square brackets hold arrays

In JSON, values must be one of the following data types: a string, a number, an object, an array, a Boolean, null.

JSON is like XML because

- Both JSON and XML are "self-describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

JSON is unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

The biggest difference is that XML must be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

3.2.7.2 Plugins versus Python Scripts

A Python script is a set of commands included in a file that is intended to be run similarly to a program. The concept is that the file will be run or performed from the command line or from within a Python interactive shell to perform a particular activity.

Python files can be also organized in nestled modules to perform different functions with a single call. User has the control step by step of the performed operations.

Plugins are instead third part pieces of software on which the user has not the full controls as well as debug possibilities. Bugs can be present inside.

Plugins are diffuse in the open-source community (see GitHub community) since they allow to write piece of code with the minimum effort.

Some applications as well as influxDB (www.influxdb.com) allows the user to provide source data in both ways, but for advanced users the python script option is always suggested.

3.3 Data processing and integration

This chapter reports the work done in PREDIS Subtask 7.5.2. In addition to the progress reported here, this subtask handled the integration of the whole data management framework within Task 7.5 and its partners, as well as to other relevant tasks of PREDIS project.

3.3.1 Data preprocessing

This section introduces some preprocessing steps needed in data processing. The introduction is generic because each model development and prediction generation have their own special requirements. Model development and predictions (including machine learning based surrogates) were topics in Task 7.4 with their own deliverable reports. However, some preprocessing steps may be alternatively implemented and integrated directly into the data management framework. So, some technologies and libraries are also introduced. In addition, the Azure cloud computing environment was tested to be suitable for performing these operations, and simple trials were done with Azure AI

studio. However, more focus was needed on the other parts of the data-driven workflow: data collection, integration of the data management framework subsystems, and administration of the different development platforms.

The goal of data preprocessing in predictive maintenance is to clean, transform, and prepare the raw data from sensors, equipment, and other sources for effective use in predictive models. The process is often iterative. As new data becomes available, the models may be retrained and refined to adapt to changing conditions and improve accuracy. Potential data preprocessing needs for condition monitoring data include:

1. Data Integration
2. Data Cleaning
3. Time Series Data Handling
4. Handling Time-Dependent Sequences
5. Labelling
6. Data Transformation

Data integration means connecting information sources and data. Especially in condition monitoring systems, combining data from different sensors and sources is important for analysis and can improve the accuracy of condition monitoring. By combining information, new and more precise insights are often achieved.

Data Cleaning means, for example, processing missing values, detecting outliers in measured values by replacing missing data or removing incomplete data. For example, the measurement data of the sensor may contain noise that can be filtered, and the detected incorrect values can be removed to improve the quality of the data.

As already stated in health monitoring applications, data comes from many different sources at different frequencies and is timed with variable regularity. In this case, the time series data must be aligned in order for the analysis to be meaningful. This can mean, for example, resampling the data at a regular interval so that the data is consistent. Sometimes the data can have interesting features such as trends or other regularities. There are many key figures for detecting these, such as moving average, standard deviation.

Processing time series data means, for example, time alignment. Especially, if the data comes from several sensors or sources, then the time must be converted to a common date representation into appropriate date-time formats and, in addition to this, handling missing values and outliers in measured values.

Data labelling means that informative labels are added to the data, which serve as information for machine learning. Labels can indicate important information about the data, such as anomalies. Anomalies are often central and interesting in condition monitoring. The process can be automated to some extent.

Data Transformation means, e.g., scaling and normalization and feature engineering. Different data sources produce data with different scales. Normalization means that the data is scaled in such a way that they are brought to a common scale (i.e., mean removal and variance scaling). Feature engineering, on the other hand, means extracting interesting information from raw data and converting it into a format understood by machine learning models.

Data preprocessing libraries for Python

Numpy (https://numpy.org/), a library of numerical operations, has been made for the Python language, and many other libraries are built on it. It supports data manipulation based on arrays and

matrices. Pandas (https://pandas.pydata.org/) is a easy to use and powerful data analysis library. It provides data structures such as DataFrames that make data cleaning, transformation, and analysis possible. SciPy (https://scipy.org/) is based on NumPy and extends for scientific and technical computing modules. In terms of condition monitoring, libraries related to signal processing and statistical analysis are useful. VTT's O&M analytics is a python library specialized on machine diagnostics, prognosis, and predictive maintenance. It is built on numpy/scipy. Scikit-Learn (https://scikit-learn.org/stable/): Scikit-Learn is widely used machine learning library. It is aimed at generic machine learning tasks, such as classification, regression, etc. It has a comprehensive set of tools for preprocessing, feature selection, and modelling. Due to its generic nature, it is also well suited for condition monitoring applications. Matplotlib is a comprehensive data visualization library. With it, you can create static, dynamic, and interactive visualizations. Seaborn is based on Matplotlib and provides a higher-level interface for making visualizations. In condition monitoring and forecasting, more complex methods such as deep learning and neural networks are often used, which can be used to model increasingly complex phenomena. Libraries such as TensorFlow and PyTorch (https://pytorch.org/) have been made for these. They are widely used and have a large community.

As for time series analysis, Statsmodels (https://www.statsmodels.org/stable/index.html) is a statistical analysis library, provides a comprehensive set of tools for statistical computing, such as time series analysis. Other libraries that specialize in this are Prophet, tsfresh, and tsflex.

Tsfresh (https://tsfresh.readthedocs.io/en/latest/) is a Python package that enables the extraction of time series features from data for analysis or machine learning use by simplifying automatic feature extraction.

Featuretools is an open source Python framework that enables the use of temporal and relational data designed for automated feature design.5

Comparison of the Python libraries

1. Functionality
    a. Numpy and Pandas are well suited and versatile for generic data processing.
    b. Scikit-Learn is well suited and versatile for generic machine learning tasks.
    c. TensorFlow and PyTorch specialize in neural networks and deep learning.
    d. Prophet, and tsfresh are libraries specialized for time series analysis.
2. Licensing
    a. VTT's O&M analytics library is proprietary (VTT)
3. Time series analysis
    a. Statsmodels, Prophet, and tsfresh are well-suited libraries for time series.
    b. Scikit-Learn is also suitable for analysing time series and feature engineering of time-related features.
4. Flexibility and customizability
    a. Scikit-Learn, TensorFlow, and PyTorch are very flexible and applicable to complex machine learning tasks.
    b. Statsmodels is well suited for complex statistical analyses.
5. Ease of use

   a. Pandas and NumPy are widely used and well known.

   b. Scikit-Learn's API is claimed to be consistent which makes it easier to use in long run.

   c. TensorFlow and PyTorch are more complex in their applicability.

6. Documentation

   a. Numpy, Pandas, and Scikit-Learn libraries have extensive documentation.

   b. TensorFlow and PyTorch are widely used in deep learning and have a large community that could be helpful.

7. Data visualization

   a. Matplotlib and Seaborn are specialized for data visualization and can be customized to a wide range of requirements.

8. Feature extraction

   a. Featuretools, tsfresh, and tsflex (https://predict-idlab.github.io/tsflex/) are libraries for automated feature engineering.

9. For Visual inspection/monitoring

   a. OpenCV is a library specialized in computer vision and image analysis.

The aforementioned data processing libraries are related to varying degrees to generic and more specialized methods. The list is not all-inclusive, but they already largely cover the needs and requirements related to condition monitoring. In addition, the list can be supplemented by implementing additional features for e.g. generic libraries, such as in VTT O&M Analytics.

3.3.2 Automatic Azure configuration

The data management framework and the integration of components were based on Azure platform as described in section 3.1. For a fluent and efficient set up and maintenance the platform and its resources are managed by Azure CLI scripts. This section introduces these scripts, Azure Resource Manager (ARM) templates and their benefits for an infrastructure deployment.

Azure resource management can be done using scripts or Azure portal. Using scripts and Azure Resource Manager (ARM) templates facilitates automation and resource maintenance in Azure. In the context of a broader toolchain and automation process, scripts and ARM templates work together to achieve end-to-end automation and infrastructure management. One of the advantages is integration with CI/CD (Continuous Integration and Continuous Deployment). Both scripts and ARM templates are essential in CI/CD pipelines. Scripts can be used to automate many tasks in continuous development, and ARM templates are used to determine infrastructure deployment. Another benefit is ensuring security. Both scripts and ARM templates can enforce security definitions and compliance checks. This ensures that security and compliance requirements are met. As for the wider toolchain and automation strategy, scripts and ARM templates are integral to this whole. They provide an approach to more efficient and consistent management of resources and infrastructure as part of an automation ecosystem in a wider context.

In managing Azure resources and infrastructure, there are potentially many benefits when combining the use of both scripts and ARM templates. With the automation of tasks by scripts, manual work and thus also the risk of human errors is reduced. Scripts can be customized and extended to meet specific requirements, allowing for custom resource management and configuration. Resource management is faster using scripts than using Azure portal, which shortens development and

deployment cycles. There is a cost to maintain Azure resources, even when the resources are not in any use. Scripts can make it easier to reduce costs as they can be used to shut down or remove unnecessary resources.

Example of CLI scripts:

```
#params file
OWNER=”Owner Name”
RG_OWNER=$OWNER
PROJECT_END_DATE=31.8.2024
RG_OWNER_TAG=”Owner=$RG_OWNER”
RG_PROJECT_END_TAG=”project end date=$PROJECT_END_DATE”
RG_TAGS=(“$RG_OWNER_TAG” “$RG_PROJECT_END_TAG”)
RG=Rg

VM_OWNER=$OWNER
VM_OWNER_TAG=”Owner=$VM_OWNER”
VM_UPDATE=”auto-dev”
VM_UPDATE_TAG=”Update=$VM_UPDATE”
VM=Vm
VM_ADMIN=admin_user_name
VM_SKU=Standard
VM_IMAGE=Ubuntu2204
VM_TAGS=(“$VM_OWNER_TAG” “$VM_UPDATE_TAG”)

ORGANIZATION=PREDIS_ORGRANIZATION
PURPOSE=Test
LOCATION=westeurope

#composing
NAME=EuwDev”$ORGANIZATION””$PURPOSE”

#composing
RG_NAME=”$RG””$NAME”
VM_NAME=”$VM””$NAME”

STORAGE_NAME=”storagename=PREDIS_storage”
STORAGE_SKU=”storageSKU=Standard_LRS”

STORAGE_PARAMS=(“$STORAGE_NAME” “$STORAGE_SKU”)
VM_PARAMS=(“vmname=PREDIS_VM”)
```

Example of configuring resources with CLI commands (creating resource group):

```
#!/bin/bash

source ./params

echo “Creating resource group: $RG_NAME to $LOCATION”
az group create \
    --name $RG_NAME \
```

```
    --location $LOCATION \
    --tags "${RG_TAGS[@]}"
```

Creating a virtual machine:

```
#!/bin/bash
source ./params

echo "Creating VM: $VM_IMAGE"
az vm create \
    --resource-group $RG_NAME \
    --name $VM_NAME \
    --image $VM_IMAGE \
    --public-ip-sku $VM_SKU \
    --admin-username $VM_ADMIN \
    --tags "${VM_TAGS[@]}" \
    --generate-ssh-keys
```

The use of ARM templates offers the several benefits. ARM templates allow infrastructure to be defined and managed as code, which enables version controlling and ease of sharing. Templates enable easy replication of environments and resources. This is useful for development, testing, and recovery. It is also possible to decompose large and complex ARM templates into smaller reusable components. From these, you can create new configurations for different requirements as needed. This facilitates the management of complex infrastructures in particular. With the help of ARM templates, the values can be parameterized, which makes it easy to modify the templates to different situations and requirements. ARM templates offer an audit trail of changes, enabling tracking changes and approvals made to resources, what changes were made, and their timestamps.

An example network resource configuring with ARM template:

```
1)          {
2)              "name": "[parameters('networkSecurityGroupName')]",
3)              "type": "Microsoft.Network/networkSecurityGroups",
4)              "apiVersion": "2019-02-01",
5)              "location": "[parameters('location')]",
6)              "properties": {
7)                  "securityRules": "[parameters('networkSecurityGroupRules')]"
8)              },
9)              "tags": {
10)                 "Owner": "Owner Name",
11)                 "Project End Date": "31.8.2024",
12)                 "Update": "auto-dev"
13)             }
14)         },
```

This project used scripts to manage Azure resources. Command Line Interface (CLI) scripts and Azure Resource Management (ARM) templates were written based on the original resource definitions made using the Azure portal interface. In addition, ARM models were defined for desired complex resource configurations such as Virtual Machine for InfluxDB. Scripts and ARM models are stored and commented in the Gitlab repository. The following steps were taken in the development process:

1. Required resources selection (e.g., virtual machines, databases, data factory and pipelines…)
2. Azure resource settings (e.g., resource groups including location and tags etc.)
3. Networking (e.g., virtual networks, subnets)
4. Authentication and Authorization
5. Data Storage and Backups
6. Scaling and Load Balancing

The aim is to make setting up similar systems easy with the help of these already existing scripts. Based on the requirements of the corresponding system, the configuration can be changed. Only a partial example of such a system and its configuration is provided here, as configuration is a vast and complex process.

1. Environment
    a. Access to Azure subscription
    b. Access to Gitlab/Github repositories for scripts and ARM-templates
2. Clone scripts from repositories
3. Deploy resources using CLI commands and ARM templates
4. Configure resources
5. Test and verify

The installation of the corresponding system proceeds as follows. First, it is necessary to prepare the environment (i.e., access to azure subscription and gitlab/github repository). The necessary scripts and ARM templates are cloned from the repository. Resources are set up with CLI commands and ARM templates. After that, the set up resources are configured according to the needs of the new system. At this stage, testing and verification is important.

3.4 Decision framework

This chapter reports the work done in PREDIS Subtask 7.5.3 by partner IFE on the decision framework system. A decision framework refers to principles, processes, and methods to proceed from available information and needs to choices that inform actions and outcomes. The processes at radioactive waste storage are procedures-in-series where tasks and options must be selected, and this selection will be more efficient when supported in the decision framework.

3.4.1 Design and functionalities

The decision framework looks at ways to support decision-making by presenting information. This includes:

- 3D Analysis where waste containers and their properties are visualised in 3D.
- Dose Analysis where the reports from scenario planning of work performed in waste facilities are presented.
- Online analytical processing (OLAP) where waste data is analysed from different points of view.
- Digital Twin view showing results from digital twins.
- Optimization of container placement to reduce dose-rates
- Dashboards where information for different sources are gathered and customized for different stakeholders.

3.4.2 Implementation and technology selection

As part of the project, a prototype for a decision platform was developed. This prototype serves as a practical testing ground to evaluate the key components and features of a decision-making system. The platform was developed as a web site with a custom backend. Using a web site have the advantage of being easy to share without any need for installing any custom software. It's also flexible in what is possible to create compared to specialised visualization solutions.

The platform uses Typescript and JavaScript both on the backend end and frontend. Various open-source libraries were used, and the backend runs on Node.js.

3.4.3 Dashboards

A multiply dashboards (Figure 10) were created to showcase information from diverse viewpoints. The aim was to provide a personalized and targeted presentation of data, ensuring that users could access and interpret information in a manner that aligns with their unique requirements and preferences. A top-level dashboard has information about the sites including summaries of aggregated IoT data, charts from the OLAP tool breaking down the number of drums per site and waste type as well as alarms and a map showing the locations of the sites. A site and building level dashboard have mostly the same information as the top level but are filtered on the selected site or building. A container-level dashboard shows information including charts of the past temperature, pressure and humidity and a 3D view of the placement of the container within the storage grid.

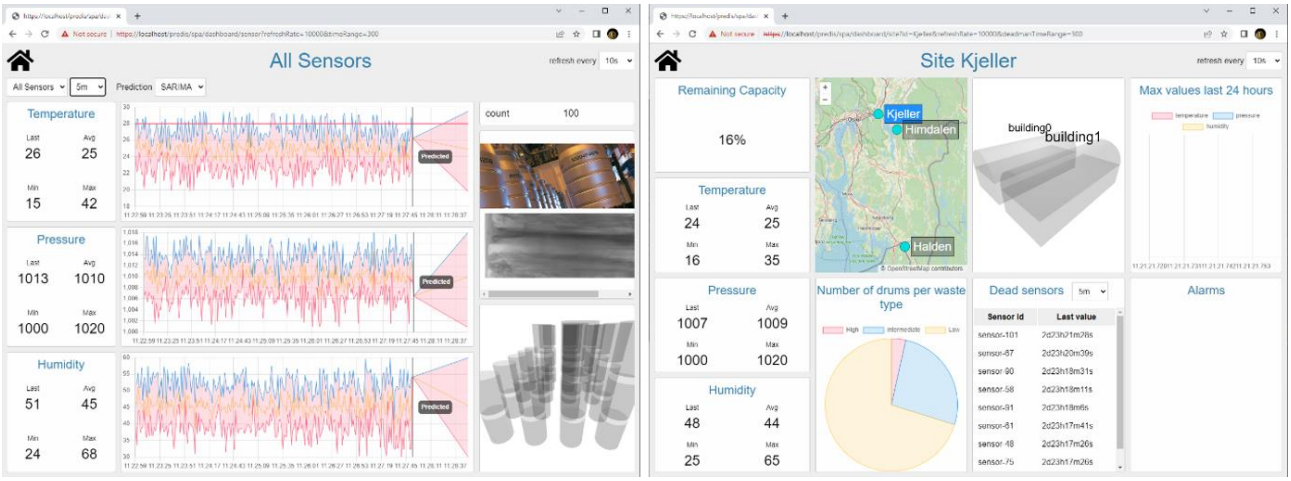


Figure 10. Example dashboards. Dashboard for a single drum on the left and for a whole site on the right.

3.4.4 OLAP Analysis

OLAP (Online Analytical Processing) analysis is a powerful data analysis technique that allows users to explore and interrogate multidimensional datasets with great flexibility. It enables the dynamic aggregation, disaggregation, and pivoting of data across multiple dimensions, such as time, geography, or product categories, providing a comprehensive and intuitive view of the information. OLAP analysis is particularly valuable for decision-making and business intelligence, as it empowers users to quickly extract insights, identify trends, and make data-driven decisions by slicing and dicing data in a way that best suits their analytical needs. This multidimensional approach is widely used in diverse fields, from finance and marketing to healthcare and environmental monitoring, to uncover patterns and relationships within complex datasets.

For the PREDIS decision platform a concept prototype was developed to show the possibility of using OLAP analysis of radioactive waste. The concept aimed to enhance decision-making by collecting and analyzing data from all waste facilities within an organization.

Since real data was not available data was generated. It consisted of 490 containers divided into 3 locations each having between 2 and 6 buildings. Time series data for temperature, pressure and humidity was generated for each container. In addition, each container was assigned a waste type to demonstrate that additional data about the containers could be analysed. Furthermore to historical data, real time data was generated for realism and to see how the system would work in a more realistic setting. Here sensors were simulated and sent through an MQTT broker to an InfluxDB

server through Telegraf, a server bases agent for collecting and sending IoT metrics. The InfluxDB was then queried for relevant data.

In this prototype (Figure 11) temperature, humidity and pressure can be aggregated using min, max or count functions. In addition, the data can be filtered or broken down by time, location or type. The data can be visualized using charts, like bar and pie chart, or the values can be shown in tables. Ad-hoc analysis is supported by using drill-down, drill-up, slice and dice operation in with immediate response.

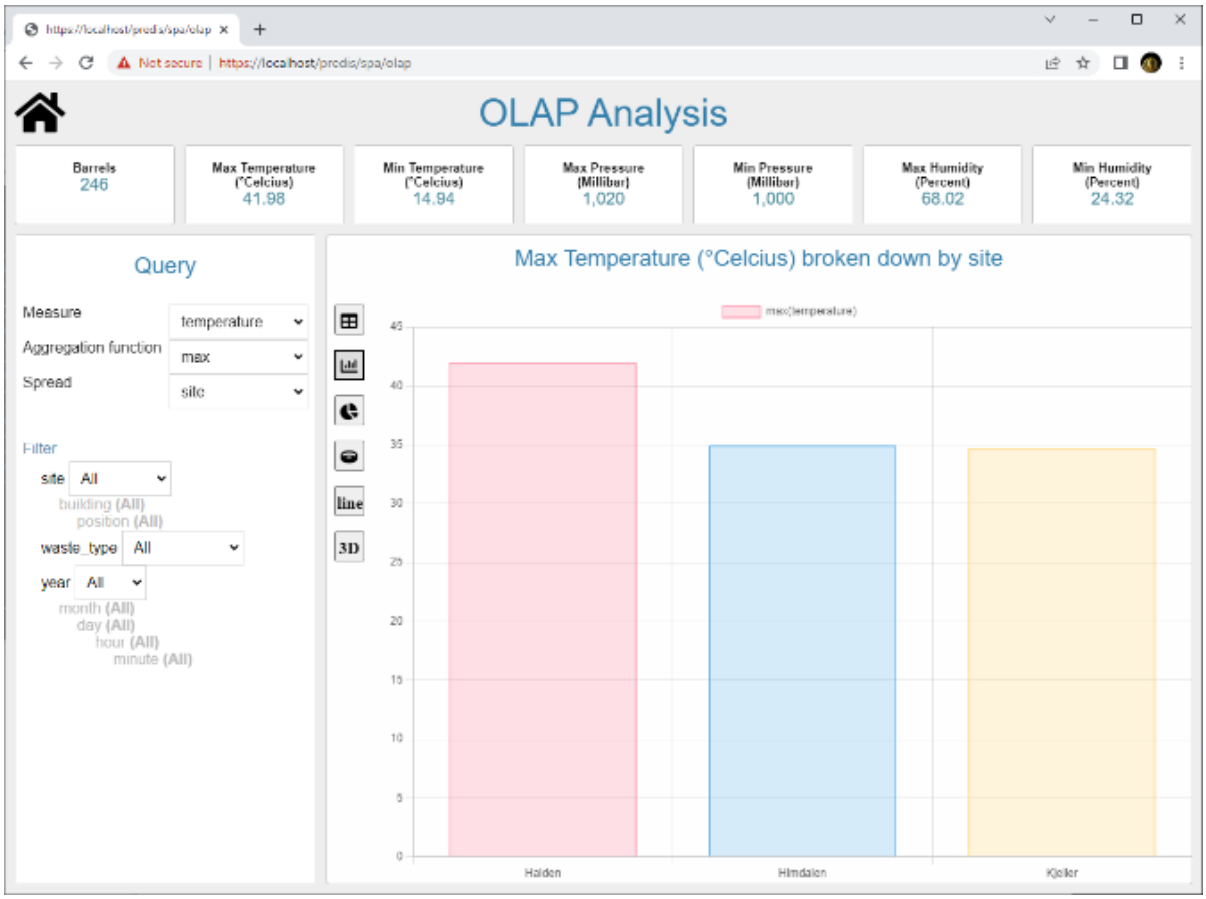


Figure 11. OLAP prototype allowing ad-hoc analysis of waste containers.

3.4.5 Dose Analysis

Waste storage facilities are potential hazardous environments due to radioactive areas. ALARA principles stipulate that received doses should be kept as low as reasonably achievable. There exist tools that can be used to reduce doses by creating, optimizing and analysing work procedures. This can be done for example by reducing the time spent in high dose areas or introduce shielding.

One such tool is HVRC Vrdose https://ife.no/en/service/hvrc-vrdose/, a 3D ALARA (As Low As Reasonably Achievable) Planning & Briefing Software developed by IFE (HVRC Vrdose – IFE) . This software allows the creation of a radiological scene that dictates the calculation of dose rates, employing methods such as the point-kernel technique with sources and shields or interpolation of measurements. A dynamic work procedure is designed, akin to an animation, defining the positions of all workers and moving components. Vrdose provides comprehensive reports summarizing scenario results, encompassing scenario duration and dose updates, both collectively and segmented by workers.

Within the decision platform, these Vrdose reports are readily accessible (Figure 12), viewable through the same web interface, facilitating convenient access to dose simulations that prove invaluable for informed decision-making.

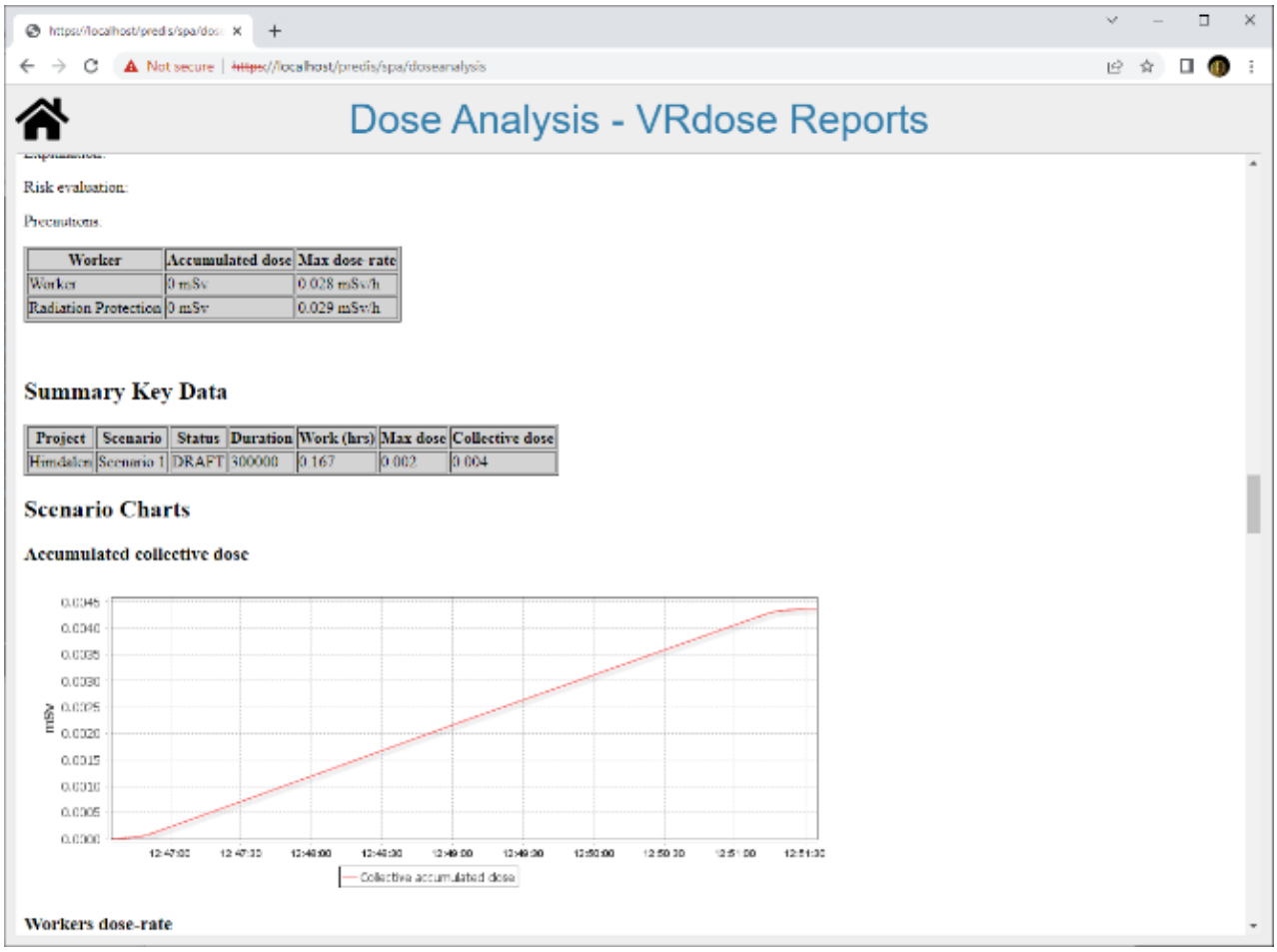


Figure 12.Example of a Vrdose report that is accessible from the decision platform.

3.4.6 3D Analysis

A 3D analysis part of the prototype was created to show some possibilities of 3D visualisation in a decision framework. The visualisation has two levels:

1. Visualisation of waste storage site(s).
2. Visualisation of waste containers (for simplicity drums was used in the demo).

3.4.6.1 Visualisation of site(s)

If there are more than one waste storage sites, they can be easily switched between by using a dropdown list (Figure 13). By clicking on the selected building the waste containers in that building are visualised.

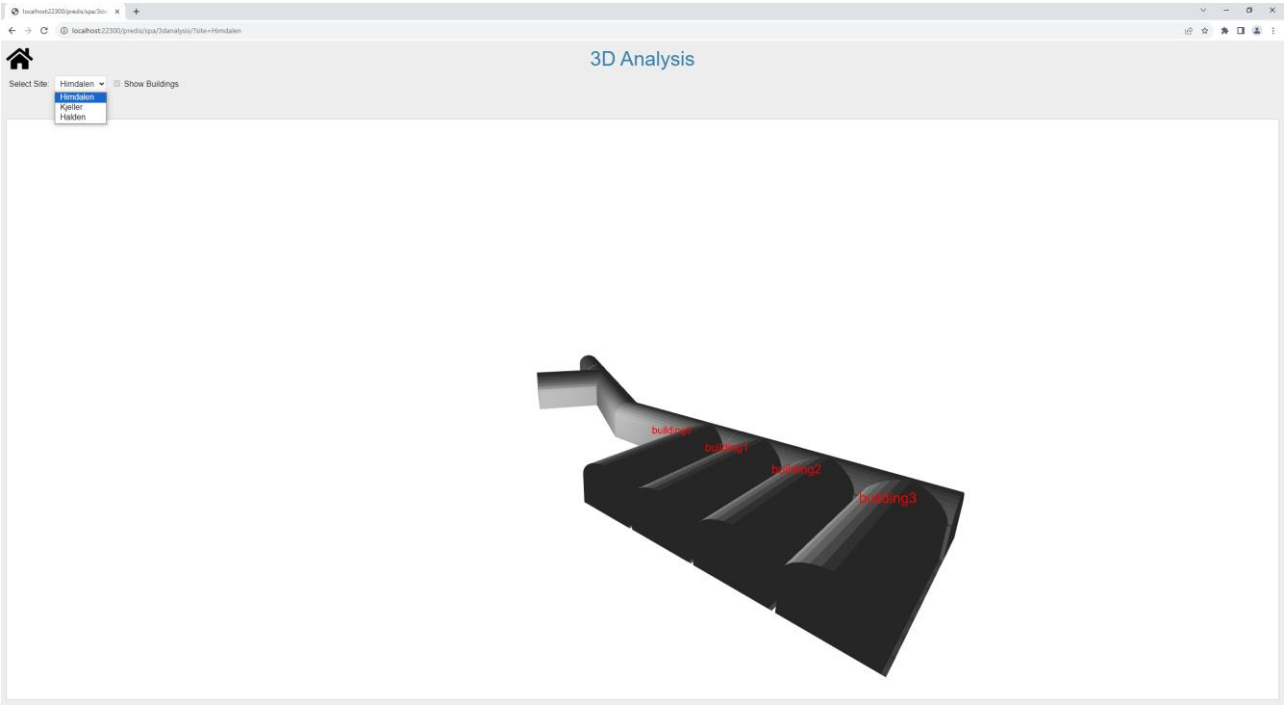


Figure 13. Visualization of a waste storage site.

3.4.6.2 Visualisation of waste containers

Here the user can switch between different visualisations of the waste containers (Figure 14). The implemented visualisations are as follows:

- Waste type (Colour coded, e.g., high level waste is red. Intermediate level waste is orange and low-level waste is yellow).
- Low-level waste (Highlights the containers with low level waste)
- Intermediate-level waste (Highlights the containers with intermediate level waste)
- High-level waste (Highlights the containers with high level waste)
- Temperature range (Colour coded, e.g., high temperature waste is red. Medium temperature waste is orange and low temperature waste is yellow).
- Maximum temperature (Highlights the container with the highest temperature)
- Minimum temperature (Highlights the container with the lowest temperature)
- Temperature limit (Using a slider the user can visualise the containers with a temperature higher than the selected value)
- Pressure range (Colour coded, e.g., high pressure containers is red. Medium pressure containers is orange and low pressure containers is yellow).
- Maximum pressure (Highlights the container with the highest pressure)
- Minimum pressure (Highlights the container with the lowest pressure)
- Pressure limit (Using a slider the user can visualise the containers with a pressure higher than the selected value)
- Humidity range (Colour coded, e.g., high humidity containers is red. Medium humidity containers is orange and low humidity containers is yellow).
- Maximum humidity (Highlights the container with the highest humidity)
- Minimum humidity (Highlights the container with the lowest humidity)
- Humidity limit (Using a slider the user can visualise the containers with a humidity higher than the selected value)
- Minimum activity (Highlights the container with the lowest activity level)
- Maximum activity (Highlights the container with the highest activity level)

- Activity limit (Using a slider the user can visualise the containers with a activity higher than the selected value)
- Minimum pH (Highlights the container with the lowest pH)
- Maximum pH (Highlights the container with the highest pH)
- pH limit (Using a slider the user can visualise the containers with a pH higher than the selected value)
- All (Show all containers)

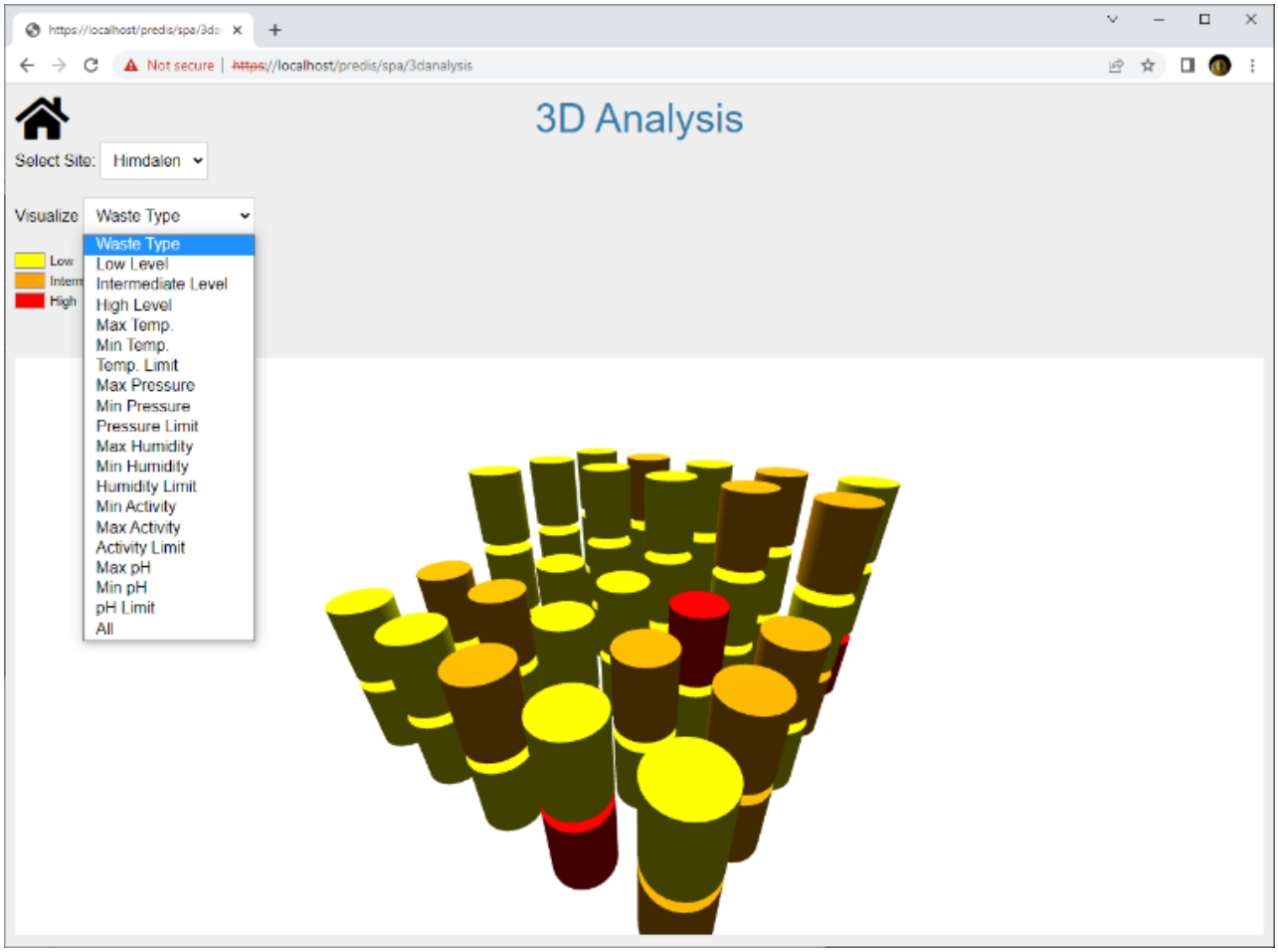


Figure 14. 3D visualization of the waste types within a waste storage facility building.

3.4.7 Optimization

The precise location for storing waste packages within a facility holds significance due to regulatory requirements and ALARA concerns. Placing high-activity containers at the center of a grid can effectively reduce the dose rate in the surrounding area, leveraging the shielding effect provided by neighbouring packages. As part of the PREDIS project, we have developed an optimization prototype (Figure 15) with the primary objective of minimizing the dose rate in the vicinity of the packages by optimizing their placement. It's worth noting that these algorithms can be extended to incorporate additional rules. For instance, certain containers, based on their contents, must not be placed adjacent to each other, or should maintain a minimum separation.

In the absence of real data when creating the prototype, we relied on generated data. The user has the flexibility to specify the number of waste packages and configure one or more rooms, each featuring a 3D grid of potential locations. Subsequently, containers with random activity levels are generated and positioned within the grids, with the highest-activity containers strategically placed at

the center. The final refinement of container placement is accomplished through the utilization of a Tabu search algorithm.

Tabu search is a heuristic optimization algorithm that has proven to be effective in solving complex combinatorial and non-linear optimization problems. Developed in the 1980s, tabu search is inspired by the concept of “tabu” or forbidden moves, which prevents the algorithm from revisiting recently explored solutions. It operates by iteratively exploring the solution space and making local improvements while avoiding revisiting the same solutions, ultimately converging toward a high-quality solution. Tabu search incorporates memory structures to store and manage previously visited solutions, enabling it to escape local optima and explore a diverse range of potential solutions.

The packing algorithm commences by generating a set of potential moves, and two distinct types of moves are devised. The first type involves swapping two randomly selected packages, while the second entails swapping neighbouring packages. Subsequently, the algorithm selects and applies the best move that isn’t marked as “tabu.” A move is considered tabu if either of its involved locations was part of another move in a specified number of past iterations. The optimal move is determined based on its ability to minimize the maximum dose rate in the surrounding area. This iterative process continues, with the user having control over the duration of execution. Alternatively, it would be possible to run for a predetermined duration, a fixed number of iterations, or until a statistically stable optimal solution is reached.

To calculate the dose rates in the vicinity of the packages, a point-kernel method was employed. Here, each package is modelled as a point source with a defined Co-60 activity and is surrounded by a shield constructed of concrete to account for its shielding effect. The dose rate was sampled at around the grid at intervals matching the grid of packages.

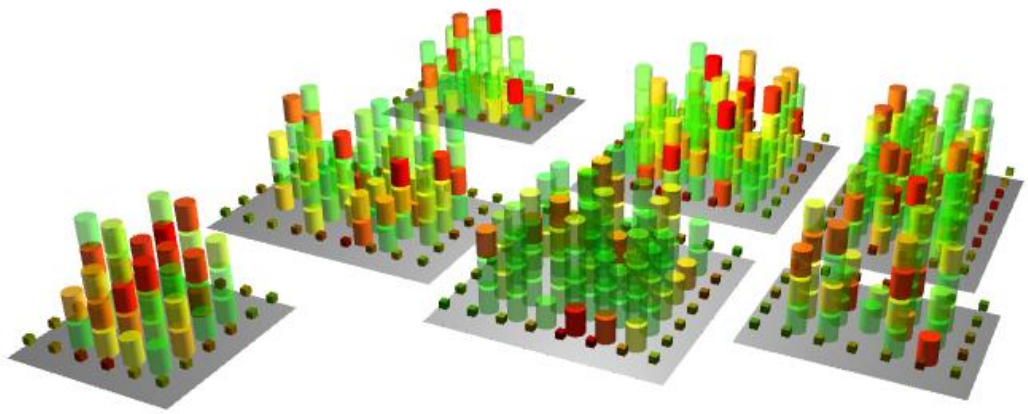


Figure 15. Visualization of an early iteration of the search. The cylinders are packages that is moved around during the search in order to minimize the dose-rate to the surrounding virtual dosimeters visualized by cubes.

3.4.8 Digital Twin integration

Parts of the decision platform is to use a digital twin to predict the values coming from the IoT devices measuring the waste containers. This concept is demonstrated in the sensor graphs shown in the container dashboard as future predictions are shown beyond the current time (Figure 16).

In addition, a user interface was prototyped to demonstrate how it would be possible to run different prediction codes from the platform. Configurations can made by selecting an algorithm and its parameters. The configuration can then be executed either once or at regular intervals. The prediction code would fetch the latest data from the database and make the predictions. The results would be uploaded back to the database. The results can be analysed on their own or as part of the historical sensor values in the container dashboard.

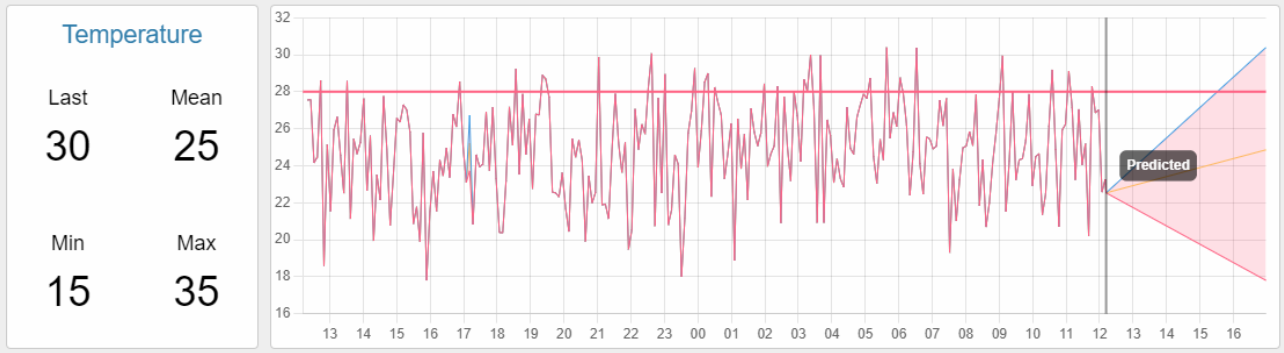


Figure 16. Future predictions from the digital twin shown on the dashboard sensor graph.

4 Summary and conclusions

4.1 Joint showcase at UJV

The development of the different subsystems of the data management framework was done by different project partners during the project in Task 7.5. To demonstrate the concept of Task 7.5 (and its link to other tasks are described in this report) as a whole, a common showcase was performed. The joint showcase demonstrates the advancements made in the project. It exemplifies the collaborative, iterative and integrative approach of the project, weaving together various strands of research and development. The specific technologies employed in this showcase from Task 7.5, were detailed in Chapter 3 of this report.

In alignment with the reference packages defined in the early stages of the project [WP7 Task 7.2.2. Reference Package and Factors Affecting Package Evolution and Degradation], selected technologies were tested in realistic environments in two different storage configurations at NNL (UK) and UJV (Czech Republic) using a set of mock-ups.

Task 7.6 focused on testing various technologies for the non-destructive testing of cemented packages in a storage environment. The objective is to evaluate these technologies to identify the most promising solutions, based on a comprehensive set of criteria defined by the end users involved in PREDIS WP7 (ORANO, SOGIN, UJV). This assessment extends beyond traditional scientific and technical considerations to include economic, safety, and operational factors, as well as training and maintenance requirements. A value assessment for the different technologies will be conducted at the project's conclusion.

The UJV demonstration (a photo of the test setup shown in Figure 17) test concentrated on validating RFID embedded sensors, SciFi gamma monitor & SiLiF neutron monitor, and sensorized RF

Identification Box for radiation monitoring (Task 7.3). The primary goal is to demonstrate the effectiveness of Non-Destructive Testing (NDT) in monitoring cemented waste packages under actual storage conditions, thereby ensuring their long-term integrity. To this end, storage conditions at the end-user facility were replicated by stacking 21 cemented packages (each 200 litres/500 kilograms) over a 3-month period at UJV Rez. The focus was on three key NDT methods developed within PREDIS Task 7.3:

- Sensorized RF Identification Box for Gamma-ray and Thermal Neutron Monitoring (UniPi).
- SciFi Gamma Monitor and SiLiF Neutron Monitor (INFN).
- RFID Embedded Sensors (BAM and VTT).

The success of this test hinges on instrumented mock-up packages designed to specific reference conditions:

- ASR Challenge with RFID Sensors: Testing RFID embedded sensor technology using Relative Humidity, Temperature, and Pressure sensors, along with Belgoprocess aggregates, to simulate conditions susceptible to an Alkali-Silica Reaction (ASR).
- Radioactivity Assessment with Cs-137 Source: Evaluating radioactivity using a mock-up with a central aperture for a 167 MBq Cs-137 source, facilitating measurement around the mock-up package and identifying anomalies or faults within cemented packages.

Figure 17. Storage configuration test at UJV Rez.

The commissioning of the mock-up facility was completed in October 2023. Preliminary data collection tests were conducted locally at UJV Rez to ensure seamless integration and data transmission to the Azure Platform. Remote troubleshooting procedures were established for any potential issues during the 3-month testing period, which concluded in January 2024.

Further, the joint showcase produced data in the form of the metadata file specified in the Chapter 3.2.3.1 of this report, and the measured data from 3 sensors from the **sensing and monitoring systems** was transferred and stored to the cloud **data platform** through IoT applications. The data was processed and displayed to the end users through the **decision framework**. Unfortunately, at the time of the writing of this report, the final data connection with **digital twin** of Task 7.4 was still under development, but the decision framework dashboard has capabilities to display the analysis data coming from the digital twin. The joint showcase is planned to be presented at the final seminar of the project in June 2024.

4.2 Conclusion

This section summarizes the main efforts of Task 7.5. Efficient and safe management of radioactive waste is of utmost importance for protecting both present and future generations. Medium to long-term storage necessitates the monitoring of waste packages for potential degradation phenomena. Managing any arising issues prior to transport to the final repositories is crucial. To ensure the safety of radioactive waste management, advanced monitoring systems need to be developed and put into operation. The common aim of WP7 tasks of the PREDIS project was to develop a concept and a prototype for a data management framework for condition monitoring system for cemented waste handling and pre-disposal storage. The objectives of Task 7.5 (as further specified in Chapter 1) are listed below, with the mention of which of the developed subsystems of the data management framework tries fulfilling the goal. The main results are also briefly summarized in this chapter.

- Conceptual Model for Data Handling and Storage – Data platform (as described in Chapter 3.2)
- Translation of NDE and Monitoring – Data management framework (as described in Chapter 3)
- Fusion of Multi-Method Monitoring – Decision framework (as described in Chapter 3.4)
- Data Integrity Plan – Data platform (as described in Chapter 3.2)
- Database and Software Prototype – Data management framework (as described in Chapter 3)

The overall aspects of data management oriented to waste management have been analysed, according to the state of the art, by investigating using dedicated case studies both the possibility to manage data the on cloud as well as on local servers and in a mix of two.

Starting from the definition of input data, which can be classified into two macrofamilies [(raw, formatted, metadata), (time series)], and of output data, once data have been stored on cloud and/or server (mainly queries for prediction / dashboarding), technical specifications have been developed and different case studies set-up to verify the feasibility of the system required. Data management for long time implies the resolution of many issues, since in the IT world any technology can become obsolete in few years (less than 10), in particular:

- Definition of life management requirements (generation and collecting of data, storing and managing data, using and sharing data between different EU organizations, archiving data, destroying data)
- Definition of reference databases trade-off analysis by considering that at least two different technologies one for metadata: SQL (Microsoft SQL, PostgreSQL, SQLite 3) and one for real time series / formatted data: NoSQL (MongoDB, RavenDB, Cassandra, BigTable, CouchDB; SQLite 3, Apache Arrow Flight SQL)
- Definition of API specification (I/O string data format: JSON, Plugins: Python scripts).

In the frame of this project a reference data-management platform (open source) has been developed by considering that:

A) users shall be able to operate in parallel on cloud systems (MS Azure, AWS) by using virtual machines and containers, on local servers (Linux based) as well as on operator workstations (Windows 10/11 based)
B) Waste can be moved from one building to another in the same facility as well as to different sites and even different countries inside the EU.

On any IT platform named in A), Miniconda (a small bootstrap version of Anaconda that includes only conda, Python, the packages they both depend on, and a small number of other useful packages like pip, zlib, and a few others) can be installed. In this environment the two Python REST API

Frameworks (Django and Pandas) in the associated virtual environment in Conda have been used for metadata (Django) and time series management (Pandas). Real time data (measure results) can be acquired both in real time and/or offline by InfluxDB (version Cloud (commercial) or Edge (open source)) and managed locally by Graphana based interfaces or queried/moved inside/outside the influxDB framework to be used in other APIs for prediction and/or decision framework (on PCs or directly on MS Azure).

Machine learning and advanced signal processing methods were identified to be more in the focus of the digital twin development of PREDIS Task 7.4, and they didn't have a strong presence in this task. Azure cloud computing environment was tested to be suitable for performing these operations, and simple trials were done with Azure AI studio. More focus was needed on the other parts of the data-driven workflow: data collection, integration of the data management framework subsystems, and administration of the different development platforms.

There are many options for modelling data. The possible use of ontologies in the context of the PREDIS project is described in Chapter 2.3. Potential benefits of ontologies, such as information integration, compatibility, and shared terminology, have also been described and considered.

Data preprocessing is presented in Chapter 3.3.1. There are many types of preprocessing methods for different purposes. Data integration, data cleaning, time series data handling, handling time-dependent sequences, labelling, data transformation were presented in this report. In addition, software libraries intended for data processing were briefly presented and compared. Here, we focused only on libraries intended for data processing of the Python programming language.

The data management framework was implemented in Azure. Automatic setup and management of this using scripts and ARM templates is presented in Chapter 3.3.2. Several advantages achieved by using scripts and ARM templates were presented. In addition, a list of steps with which a partially corresponding system can be deployed was compiled.

As part of the decision framework initiative, a prototype for a decision platform was designed and implemented as a web application. This platform incorporated multiple dashboards, each tailored to present information from distinct perspectives, tailored to the specific needs of various users. Users can access 3D analysis views that facilitate the visualization of both waste storage sites and containers. Furthermore, the decision platform offers OLAP analysis of radioactive waste and dose analysis reports (ALARA based planning or briefing option). Additionally, the platform goes beyond presenting only the current situation by showcasing future predictions derived from a digital twin, which can be conveniently viewed on a dashboard. This comprehensive approach ensures that the decision platform not only addresses immediate needs but also provides valuable insights for long-term planning and management.

A conceptual model for data handling and storage has been developed, and a practical implementation has been performed. The demonstration test at UJV confirmed that monitoring data from sensors can be automatically uploaded to the Microsoft Azure cloud platform, and it can be visualized on the decision platform. To ensure traceability, the associated metadata can also be stored, and connected with the monitoring data. A web-based decision platform with multiple dashboards has been developed to present information from different perspectives customized to different users. The functionality of the decision platform has been proven in the UJV demonstration test. Predictive numerical models of the digital twin can be integrated into the platform to further support decision-making.

REFERENCES

Bagavathiappan, S., Lahiri, B. B., Saravanan, T., Philip, J., & Jayakumar, T. 2013. Infrared Physics & Technology Infrared thermography for condition monitoring – A review. Infrared Physics and Technology, 60, 35–55. https://doi.org/10.1016/j.infrared.2013.03.006

Davies, A. 1998. Condition Monitoring Handbook of Condition Monitoring. SPRINGER-SCIENCE+BUSINESS MEDIA, B.V.

Gholizadeh, S., Leman, Z., & Baharudin, B. T. H. T. 2015. A review of the application of acoustic emission technique in engineering. 54(6), 1075–1095.

Hee-Seoung Park et al. (2022) A detailed design for a radioactive waste safety management system using ICT technologies https://doi.org/10.1016/j.pnucene.2022.104251

ISO 13374-1, 2003. Condition monitoring and diagnostics of machines — Data processing, communication and presentation — PART 1: General Guidelines. First ed.

ISO 13374-2, 2008. Condition monitoring and diagnostics of machines — Data processing, communication and presentation — PART 2: Data processing, Corrected version. First ed.

ISO 14830-1:2019. Condition monitoring and diagnostics of machine systems — Tribology-based monitoring and diagnostics — Part 1: General requirements and guideline. First ed.

ISO 17359:2019. Condition monitoring and diagnostics of machines — General guidelines. Second ed.

ISO 18434-1:2008. Condition monitoring and diagnostics of machines — Thermography — Part 1: General procedures. First ed.

Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). “A review on machinery diagnostics and prognostics implementing condition-based maintenance.” Mechanical Systems and Signal Processing, 20(7), 1483-1510

Kolditz, O. et al., 2023 Digitalisation for nuclear waste management: predisposal and disposal, Environmental Earth Sciences (2023) 82:42 https://doi.org/10.1007/s12665-022-10675-4

Mathew, A.D., Zhang, L., Zhang, S., Ma, L., 2006. A review of the MIMOSA OSA-EAI database for condition monitoring systems, in: Proceedings World Congress on Engineering Asset Management. Gold Coast, Australia. https://doi.org/10.1007/978-1-84628-814-2_88

MIMOSA, URL https://www.mimosa.org/what-is-mimosa/ (accessed 10 Mar 2023).

Mohanty, A. R., (2015). Machinery Condition Monitoring: Principles and Practices. Boca Raton: CRC Press.

OM – Ontology of units of Measure, 2017, https://github.com/HajoRijgersberg/OM (accessed 25 Jan 2024)

PREDIS Deliverable 7.1 State of The Art in packaging, storage, and monitoring of cemented wastes 2021-04-14 version 1.1

PREDIS Work Package 7 Milestone MS53 – M7.5 – Month 36 Internal Report “Prototypes ready for demonstration in real environments”

PROV-O: The PROV Ontology, 2013, https://www.w3.org/TR/prov-o/ (accessed 25 Jan 2024)

Randall, R. B. (2011). Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications. Wiley.

Rao, B. K. N. (1996). Handbook of Condition Monitoring: Techniques and Methodology. Elsevier.

Réka Szőke, Ernst Niederleithinger et al., (2023) Predisposal Management of Radioactive Waste: Digital Decision Support System, WM23 23237 Full length paper.

Semantic Sensor Network Ontology, 2017, https://www.w3.org/TR/vocab-ssn/ (accessed 25 Jan 2024)

Waqas Ashraf, 2018. A model to assess and customize computerized maintenance management systems complies with industry 4.0 vision and requirements: A case study in food processing plant. University of Stavanger, Stavanger.

Appendix 1: Data platform case studies and experiences

This appendix contains freeform descriptions of the case studies done during the data platform development by ANN. They are included for information and possibility to share some of the experiences gained during the project. They are hopefully helpful for someone interested in developing similar systems.

Case Studies

Case studies have different objectives:

a) Analyse as metadata can be managed
b) Analyse as time series can be managed
c) Verifying the coupling between metadata and time series.

The investigation should involve different approaches and it should be closed to the real situation: in particular, a data pipeline should be organized between sensors and databases to highlight the bottlenecks.

Currently the following investigations have been performed:

a) Use of “MS Azure Data Lake” Storage to save data on the long-term “on Cloud”
b) Use of Django as open-source tool to create a platform to manage metadata and HMI
c) Use of SQLite 3.0 as reference open-source database (alternative is PostgreSQL if multi-access IP is required)
d) Use of MS Azure IOT Central to manage time series data acquisition in an IoT environment
e) Use of InfluxDB to manage time series in a multi-database (i.e. buckets) structure.

Python has been defined as reference programming language.

Time Series Management

In Subtask 7.5.1, case studies have been planned in two different phases and with different objectives:

1. in a first phase data flow generation are simulated by using devices not strictly related to Task 7.3 (STM Microelectronics ROS node and ESP32)
2. in a second phase data flow generation is coming from Task 7.3 developed instrumentation.

Some constrains (i.e., preliminary design choices/assumptions) have been fixed in this research frame:

a. Data management can be allowed both on local server and on cloud
b. The waste package in only constituted of a basic unit (i.e., drum)
c. Sensor and waste package are independent unit until they are permanent associated
d. For “Time Series” final user (i.e., Subtask 7.5.3) can perform only queries on waste package (i.e., drum)

In details:

- Basic Units have to be loaded on SQL database with at least the following information:
  - a set of metadata (a tag/barcode, the waste characterization after curing, ...)
  - the planned periodic verification plan
  - the associated blockchain string to track the waste history (if any)
- Sensors are to be loaded on SQL database with:
  - a set of metadata (tag, type of measure, calibration certificate, calibration expiring)
  - indication if they are inside/or outside a waste package
  - the basic units ID for which the measure is significative
- “Sensor-node/ DAU” are to be loaded on the database with

- The sensors which are referring to,
- The firmware release
- The communication protocol and/or data format
- The calibration certificates tracking information

Users can perform queries on database by considering:

- The reference object for DB Query is the Waste Package
- For a basic unit (i.e., drum) it shall be possible to query for:
  - The drum history
  - The associated measures
  - The associated predictions
  - The periodic verification plan
  - The tracked changes

MS AZURE as Data storage on Cloud

Large-scale data warehousing and analytics involves two key elements, data ingestion and data processing.

Data ingestion is the process used to load data from various sources into a central data store.

Data can be ingested using batch processing or streaming, depending on the nature of the data source.

Data processing involves operations on the data to clean, filter, restructure, and prepare the data for analysis.

MS AZURE allows a pipeline based on

- Input data is mixture of relational and non-relational data from many different sources, batched or streamed
- Data is ingested from various sources with Azure Data Factory to Azure Data Lake on top of Azure Blob Storage
- Hierarchical Container structure (with a template) is build inside Storage to build logical segments of data
- Containers are accessed by other Azure services such as Azure Synapse Analytics, Azure Analysis Services and Azure Machine Learning Studio to process and visualize

Azure Data Factory is a data ingestion and transformation service that allows loading raw data from many different sources, both on-premises and in the cloud.

As it ingests the data, Data Factory can clean, transform, and restructure the data, before loading it into a repository such as a data warehouse.

Work performed by Azure Data Factory as a pipeline of operations

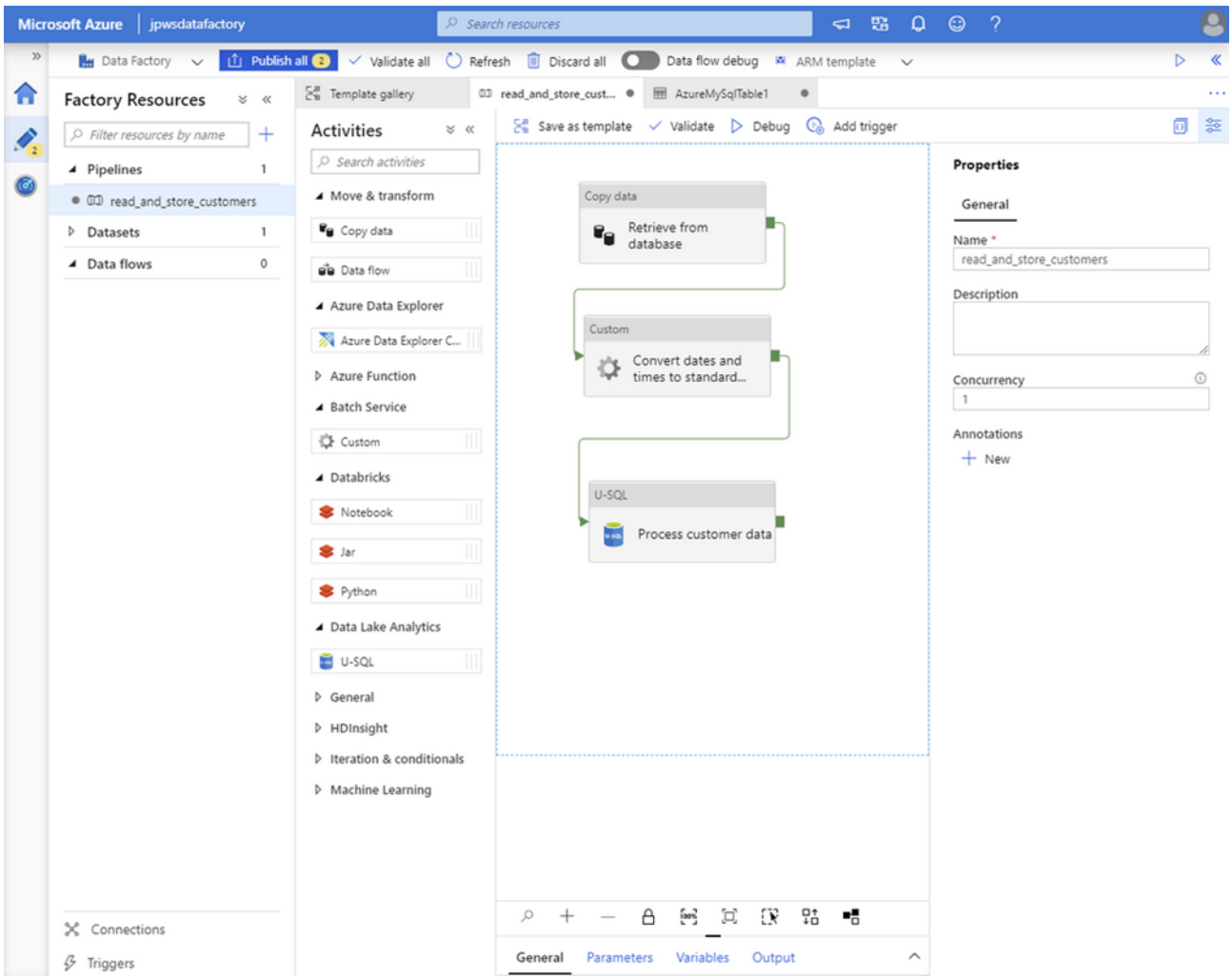


Figure A1.1. Azure Data Factory

Data Lake Store provides a file system that can store near limitless quantities of data. It uses a hierarchical organization (like the Windows and Linux file systems), but can hold massive amounts of raw data (blobs) and structured data

1. hierarchical directory structure needs to be configured: *{Partner}/{Source(/Plant)}/{Sensor/Dataset}/{yyyy}/{mm}/{dd}/* (?)
2. Enables granular Role-Based Access Control (RBAC)
3. Compatible with the Apache Hadoop

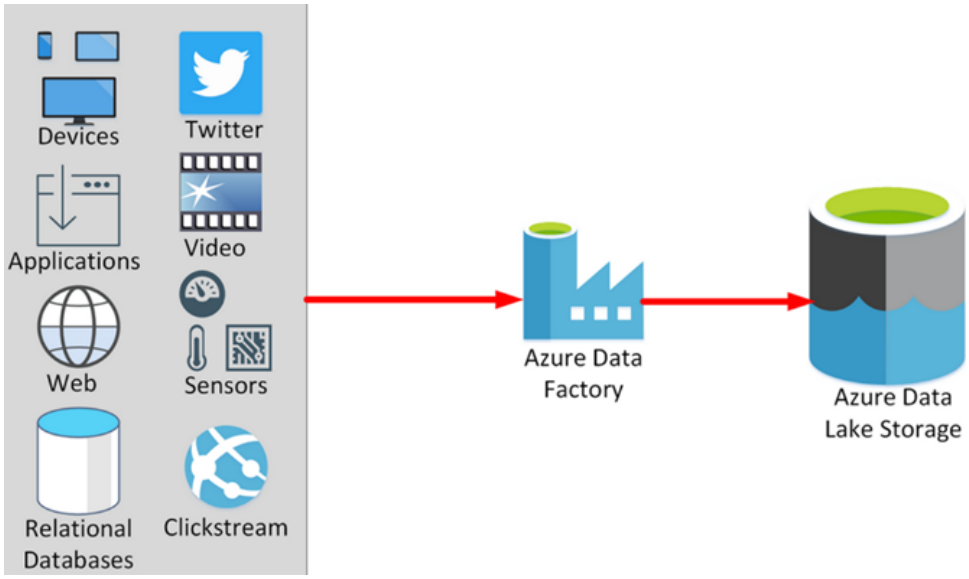


Figure A1.2. MS Azure Schemes

On the basis of above VTT's has opened a Azure Subscription for the project

1. BA4_124983-1.7.5_EU_PREDIS_WP7.5 (63d-c370-4c52-9f92-e1a831905fc1)
2. VTT works as admin + we have to follow VTT's governance rules

- Project partners can be invited to use this subscription
  3. Work emails only
  4. Similar authentication as VTT's Teams site
  5. Role-based-access rights (Owner / Contributor / Reader) on service level
- Everything for the project can be done on Azure under this subscription
  6. All costs will be billed monthly by VTT IT from the project

MS Azure Container is characterized by:

a. Access Control Policy (IAM)
b. Shared Access Token in order to give the possibility to the world outside to access to the container in order to store data or read data without accessing to the subscription and/or directly to the MS Azure portal
c. Manage Access Control List (ACL)
d. Define an access policy (i.e. who can access to the container from outside)
e. Properties related to an URL (i.e. https://euwdatahandlingstorage01.blob.core.windows.net/ansaldo)
f. Extract metadata.
g. Possibility to upload wildly files by connecting to the subscription

MS AZURE Central as Reference IOT platform

One possibility to manage on cloud data is to use MS Azure Central IoT.
Azure IoT Central preassembles platform as a service (PaaS):

- Allow quick connectivity between IoT devices and the cloud
- Centralized management to easily reconfigure and update devices
- Visualize and analyse IoT data, from the big picture to small details, by "Data Explore" and "Dashboard" function
- Bridge business applications and IoT data:
- Create "jobs" (define schedule processes by including different devices)
- Define "rules" to monitor single devices at scheduled time and trigger "actions"

- Export or send filtered data through the Cloud versus other platforms (i.e. Azure Data Lake).

Devices connected to IoT Azure are suggested to implement AZURE RTOS, which is under development by looking SIL4 applications. Starting from end 2021 MS Azure has started:

1. to certify industrial devices in order to allow “secure” and “blinded” connection according to the AZURE standards
2. to signing collaboration contracts with different industries as partner (i.e., Renesas Electronics Corporation, STM microelectronics, NXP Semiconductors N.V) in order to develop dedicated RTOS implementation to be (later on) certified. A wide catalogue of components connectable to IoT Central is already present on web (see https://devicecatalog.azure.com/featured)

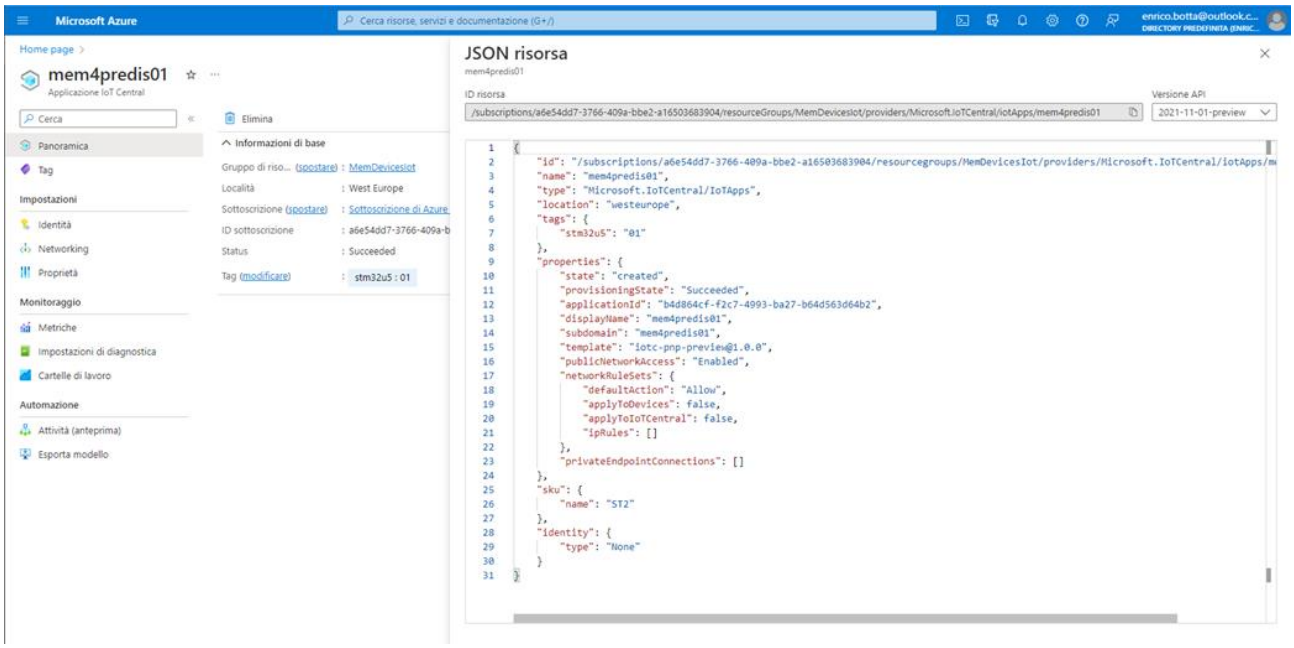


Figure A1.3. Definition of Resource on MS Azure IoT Central.

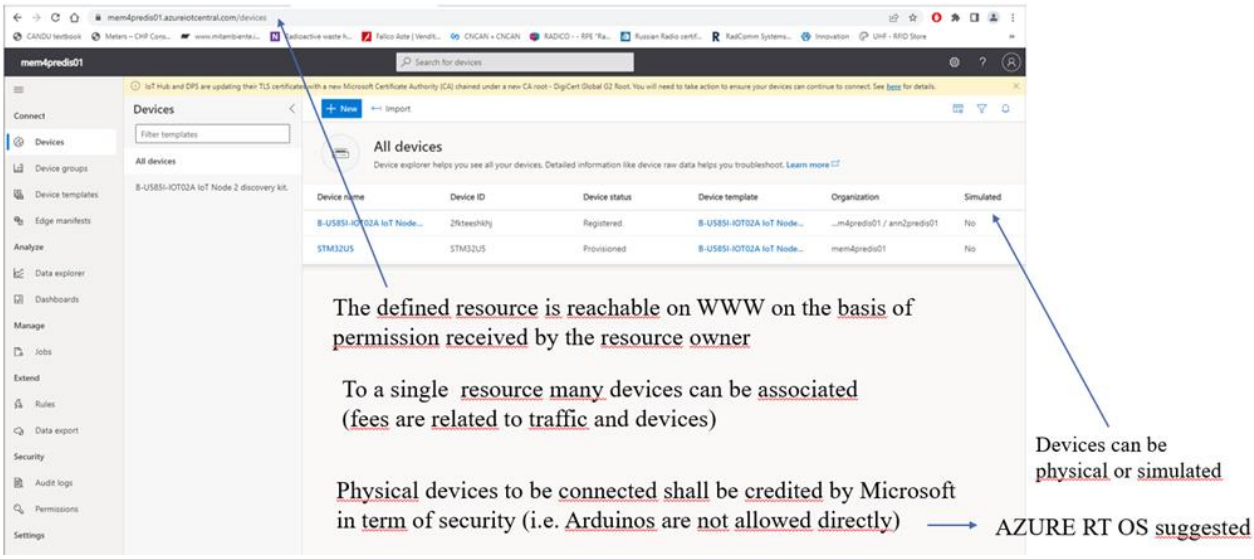


Figure A1.4. Attach devices at your MS Azure IOT Central resource (254 devices allowed).

In order to simulate 24 hours on 24 hours the measuring data associated to a single drum, data which should come from 7.3 in the next phase of project, ANN prepared a case study where:

- Input data are generated by using a last generation (2022) STM developing board as DAU (STM32U5 not commercial version) with the firmware programmable with MS RT OS (“Microsoft Real Time OS) able to generate a continuous flow of data versus MS Azure IoT Central platform according to the MS Azure security protocols.
- The reference measures to manage (by using on-board sensors) have been set in
    - Three scalar variables: temperature, pressure, humidity,
    - Two vectorial variables Vectorial: acceleration, magnetic field
- Measured data are acquired (via MQTT) in Real Time at a fixed frequency on a IoT dedicated cloud application (i.e. https://mem4predis01.azureiotcentral.com)
- The MS azure IoT application works a self-standing dashboard (up to 1e6 sensors can be connected at same time), but data can be queried locally and query procedure can also generate alarms on selected criteria: in the present case if the temperature is above an threshold value the whole string is exported in a «self-explaining» string in JSON format outside MS azure domain, in particular in VTT MS Azure BLOB Container named “Ansaldo” (i.e. MS Azure Data Lake).
- On VTT BLOB Container a list of string is archived and they can be retrieved and manage independently by the application which originates the data.

Each organization can create independently an application under Azure IoT Central and decide if other organizations can access or not.
Communication from local instruments or server requires authorized «nodes» in order to exchange data in accordance with Microsoft security protocols.
Each R&D organization can (or ask to) define a Container (and the subdirectories) in the VTT Data Lake DB
Create a pipeline between Azure IoT Central and an Azure Container by addressing the URL (i.e. https://euwdatahandlingstorage01.blob.core.windows.net/ansaldo) in the IoT Central application.
Export automatically selected data from IoT Central by defining «Alarm» Criteria

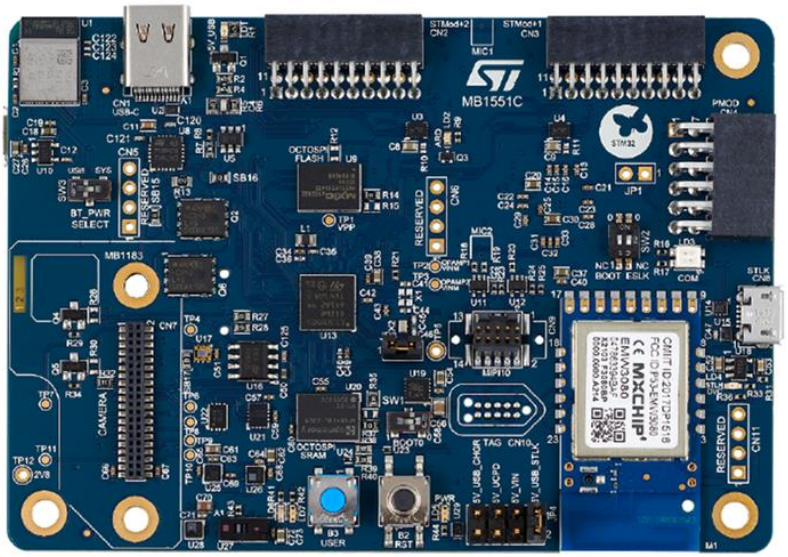


Figure A1.5. Case Study – STM ARM Cortex-M33 processor developing board with an ultra-low-power microprocessor (not commercial).

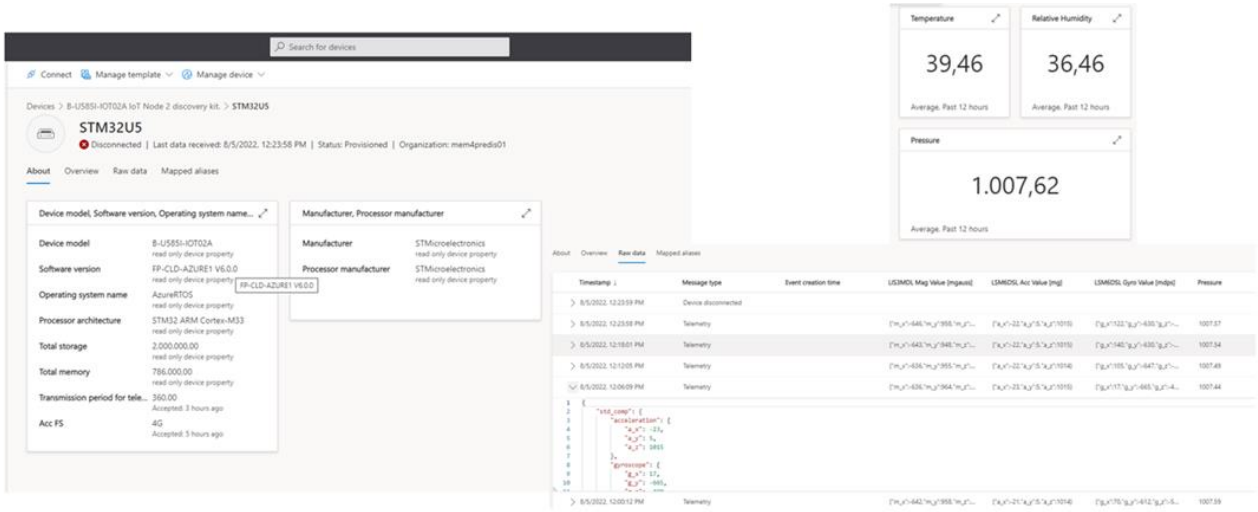


Figure A1.6. Case Study – Data Flow Simulation on MS Azure Central IoT Platform.

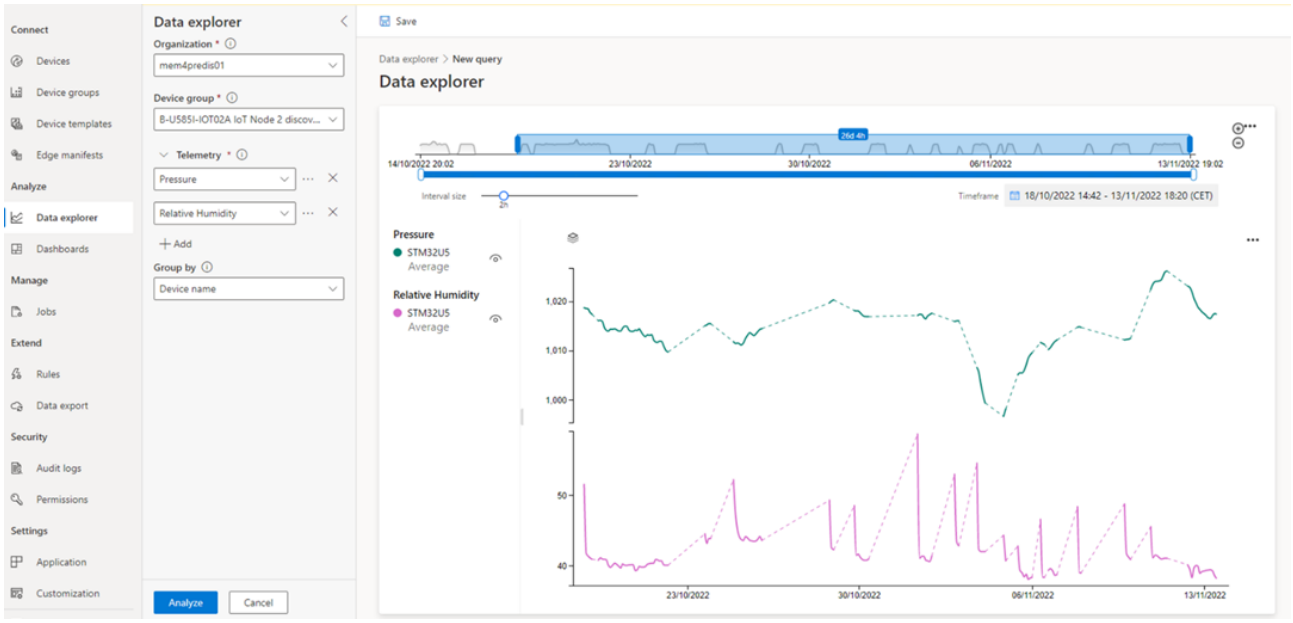


Figure A1.7. Case Study – Data Exploring on MS Azure IoT Central Platform.

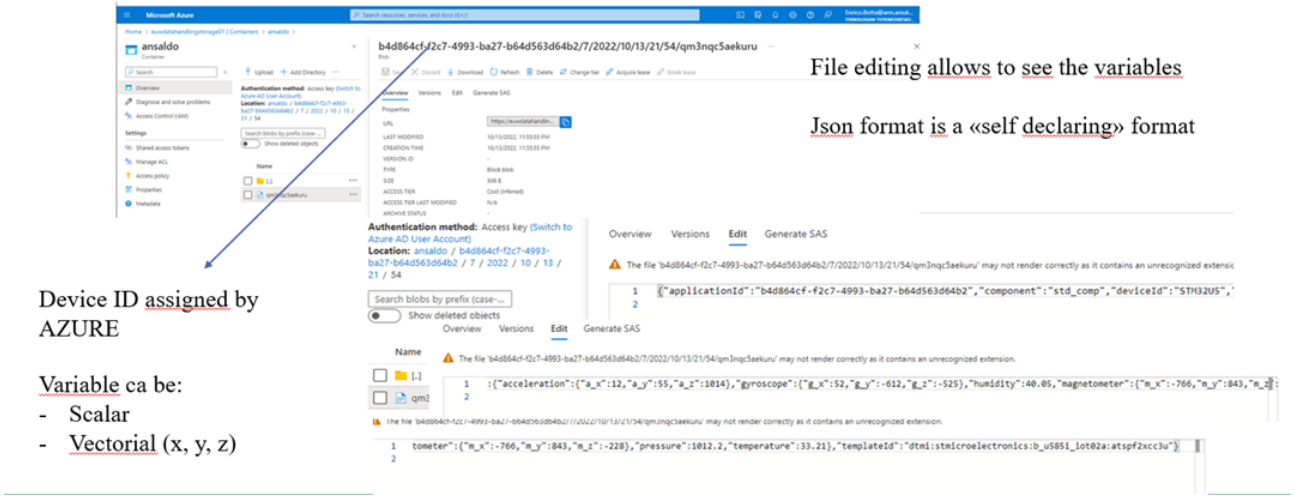


Figure A1.8. Azure Event Tracking – A Typical JSON format.

Data to be Managed in the Case Study

Data to be managed in the case study are Metadata & Time Series Data, which are two sets of different type of data to be managed with different requirements.

Requirements includes use of IT open-source SW platform with some main constraints

- All SW packages (used for developing) should be consistent with a potential migration on Cloud (AZURE, AWS)
- Sensors and DAU developed in Task 7.3 providing input data from field should be compliant with Industry 4.0 & IoT application (STM Node, ESP32, LORA, ...)
- Language to write down scripts compliant with current way to proceed (i.e., Python, Node.JS, React, Arduino, ...)
- Databases to manage both meta-data and raw-data should be consistent (i.e., SQLite3, PostgreSQL, MariaDB, etc)

Technical choices preliminary selected to manage data are:

- Real Time Series:
  - InfluxDB (www.influxDB.com) as reference tool by including associated graphics
  - Three Buckets (i.e., databases) taken as reference with different purposes: manage sensors, manage waste package
- Meta-Data:
  - Relational Database supported from InfluxDB (i.e., PostgreSQL https://www.python.org/ and/or SQLite 3 https://www.sqlite.org/index.html (InfluxDB Enterprise))
  - JSON schemas as reference to read and write data automatically from/to the data-base involved
  - Miniconda3.0 (see https://docs.conda.io/en/latest/miniconda.html) to create virtual environment where run:
    - Python (see https://www.python.org/ security version 3.9 or 3.10) the reference programming language for all modules to write.
    - Django (http://www.djangoproject.com/) to a create a full-standing working platform by including HMI

SQL Databases for Metadata

Three big independent data families need to be managed:

- WastePackage (i.e., “Concrete Drum” in WP 7)
- Sensor Data Acquisition Unit (i.e., Detector & Monitor)
- RAW data (i.e., images from muon tomography and other big files to be linked)

Two different techniques area possible to manage metadata

a) Model –View-Controller (MVC)
b) Model – View –Template (MVT) used in Django (https://www.djangoproject.com/)

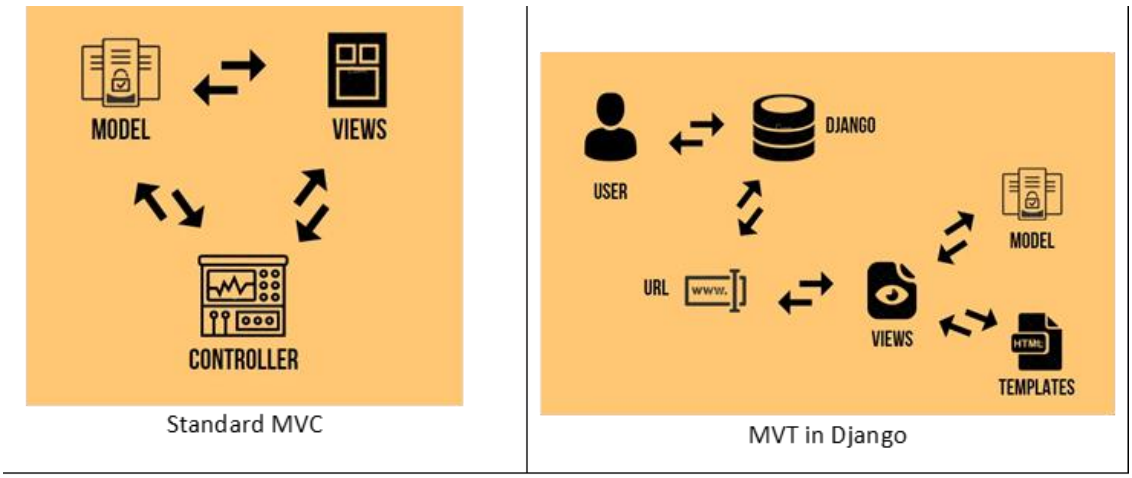


Figure A1.9. Modelling and managing metadata.

In MVC:

- A model provides the interface for the data stored in the Database. It is responsible for maintaining the data and handling the logical data structure for the entire web application
- A view is a user interface. It is responsible for displaying Model data to the user and also to take up information from the user. Views in MVC is not the same as the Views in Django
- A controller is responsible for the entire logic behind the web application. That is when the user uses a view and raises an Http request, the controller sees the user request and send back the appropriate response.

In MVT:

- A model, as in MVC, provides the interface for the data stored in the Database.
- Like views in MVC, Django uses templates in its framework. Templates are responsible for the entire User Interface completely. It handles all the static parts of the webpage along with the HTML, which the users visiting the webpage will perceive. Templates are written in Django Template Language, a simplified version of HTML.
- View acts as a link between the Model and the Templates by creating a bridge: it sees the user request, retrieves appropriate data from the database, then renders back the template along with retrieved data.

In Django MVT architecture there is not a separate controller as in MVC and everything is based on MVT.
The advantages are:

1. The user sends a URL request for a resource to Django.
2. Django framework then searches for the URL resource.
3. If the URL path links up to a View, then that particular View is called.
4. The View will then interact with the Model and retrieve the appropriate data from the database.
5. The View then renders back an appropriate template along with the retrieved data to the user.

Django models are classes that represent a table or collection in our Database.
Each model contains all the information regarding the table.
These models are stored together in Django in a file models.py in the Django App.
There can be many different models for different DB containing different information like User DB, Book DB, or any other table required by the web application.

Django officially supports the following databases: PostgreSQL, MariaDB, MySQL, Oracle, SQLite (default). The main limitation of SQLite 3 is that it cannot be addressed by a full IP address but only with a network local IP address (AAA.X.Y.Z) where AAA is fixed for all users. For a full network access PostgreSQL is suggested as alternative.
In this case study, InfluxDB task will be created to transfer data from a Django endpoint to an InfluxDB bucket.

The working environment can be setup any PC or workstation or VM on the cloud by following some elementary instructions:

- Install Conda/Miniconda on your PC/WK/VM: https://docs.conda.io/en/latest/miniconda.html
- Install Python 3.10 or 3.11 within miniconda (or upgrade it);
- Install Node.js
- Create a virtual environment where install Django (i.e., predis)
- Install all needed libraries for the project by “requirements.txt”
- Install pillow (py imaging) and whitenoise (WSGI app)
- Run Django MVT project server (or copy the demo)
- Access webserver page (if local 127.0.0.1:8000)

The reference database is a SQLite data-base (SQLite 3 .0 or above), characterized by

- Manifest typing and BLOB support.
- Support for both UTF-8 and UTF-16 text.
- User-defined text collating sequences
- 64-bit ROWIDs (no practical physical limit to the number of rows)
- Max Size: 281 Terabyte
- Max single table size (“model” in Django, “measurement” in InfluxDB): 2GB
- Accessible in Query and Write mode by Flux v 0.x (InfluxDB)
- Data Migration in PostgreSQL (server on cloud application)

In the case study Django apps defined currently are:

- wastepack (meta-data associated to waste-packages)
- sensor (meta-data associated to sensor)
- images (meta-data associated to images)

To each Django app is associated a “model” (i.e., a table in the SQLite database).
Four reference directories (“predis01”, “productionfiles”, “mystaticfiles”, “pics”) are directories to manage the server, html and css files.
In the model files (see examples)

- Class = name of the table in SQLite 3.0
- Class member list = name of the columns in the named table
- Class member list shall contain “TAG” and “Field ID” used in InfluxDB

Besides Dynamic cross link with JSON schema can be performed by Django FRAME restwork (to investigate/perform).

```
from django.db import models

# Create your models here.

class Wastepack(models.Model):
  wptag = models.CharField(max_length=255)
  wptype = models.CharField(max_length=255)
  location = models.IntegerField(null=True)
  characterization_date = models.DateField(null=True)
  loading_date = models.DateField(null=True)

def __str__(self):
    return f"{self.wptag} {self.wptype}"
```

```
from django.db import models

# Create your models here.

class Sensor(models.Model):
  sensors_id = models.CharField(max_length=255)
  sensortype = models.CharField(max_length=255)
  location=models.CharField(max_length=255)
  model_number = models.IntegerField(null=True)
  last_inspected_date = models.DateField( null=True)

def __str__(self):
    return f"{self.sensors_id} {self.sensortype}"
```

```
from django.db import models
from django.utils.safestring import mark_safe
from PIL import Image as Im                 # new

# Create your models here.

class Image(models.Model):
    title = models.CharField(max_length=20)
    photo = models.ImageField(upload_to='pics')

    def save(self):                         # new
        super().save()

        img = Im.open(self.photo.path)

        # resize it
        if img.height > 300 or img.width > 300:
            output_size = (300,300)
            img.thumbnail(output_size)
            img.save(self.photo.path)


    def image_tag(self):
        return mark_safe('<img src="/../../pics/%s" width="150" height="150" />' % (self.photo))
```

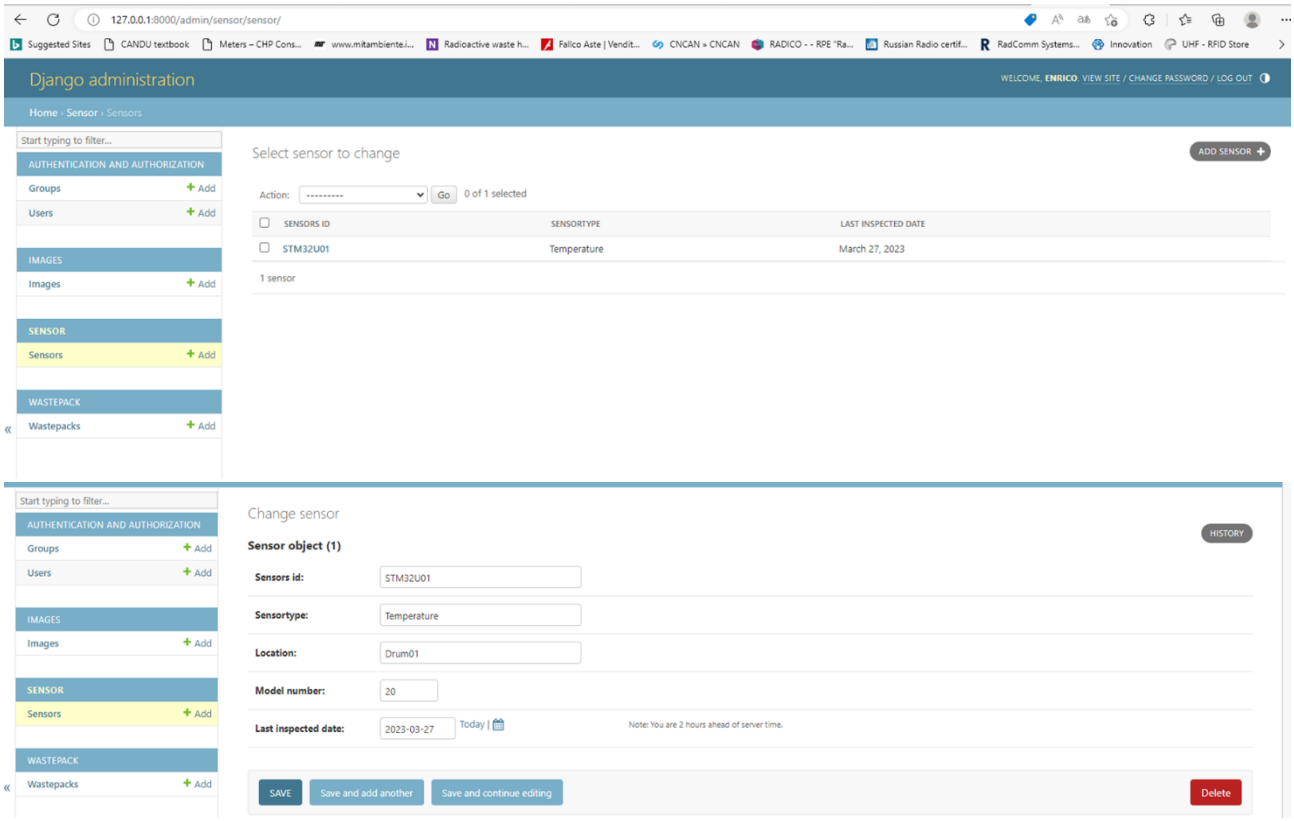
GitHub Predis-h2020 information processed with «JsonSchema2Django.Py» to check inconsistency

```
Model name is Sensor
Title of model is Waste Package Monitoring Sensor Metadata
Description of model is Metadata for a waste package monitoring sensor, includ
WARNING: Possible ForeignKey sensor_id

from django import models
from django.models import json

class Sensor(models.Model):
    """Generated model from json schema"""
    sensor_id = models.TextField(null=False, blank=False)
    pose = json.JSONField(default={}, null=False)
    product_name = models.TextField(null=True, blank=True)
    measuring_quantity = json.JSONField(default=[], null=False)
    calibration = json.JSONField(default={}, null=True)

    class Meta:
        db_table = 'sensors'
```

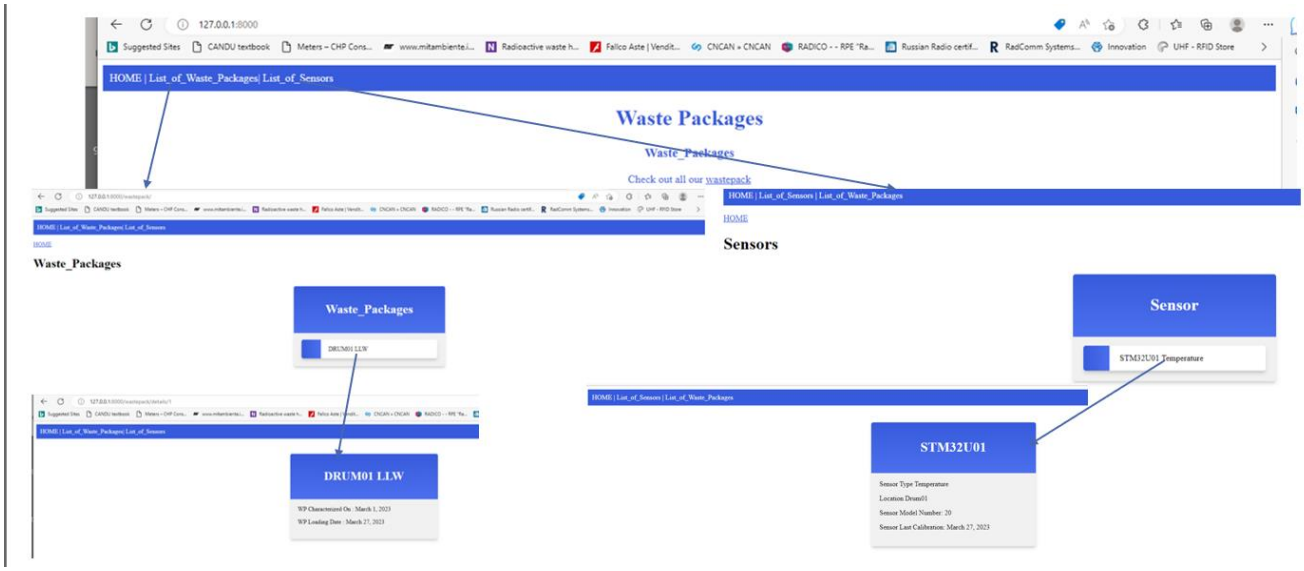
```
Model name is Unipi_lora_node
Title of model is Unipi LoRa node metadata schema
Description of model is Metadata to describe a LoRa node with up to 8 radiation sensors attached
WARNING: Possible ForeignKey drum_id
WARNING: Possible ForeignKey lora_node_id

from django import models
from django.models import json

class Unipi_lora_node(models.Model):
    """Generated model from json schema"""
    drum_id = models.TextField(null=False, blank=False)
    lora_node_id = models.TextField(null=False, blank=False)
    manufacturer = models.TextField(null=True, blank=True)
    hw_version = models.TextField(null=True, blank=True)
    fw_version = models.TextField(null=True, blank=True)
    installation_time = models.TextField(null=True, blank=True)
    channels = json.JSONField(default=[], null=False)

    class Meta:
        db_table = 'unipi_lora_nodes'
```

```
Model name is Embedded_sensornet
Title of model is Embedded SensorNet Static Metadata
Description of model is Metadata to describe a bus of SensorNodes with a pas
WARNING: Possible ForeignKey drum_id

from django import models
from django.models import json

class Embedded_sensornet(models.Model):
    """Generated model from json schema"""
    drum_id = models.TextField(null=False, blank=False)
    installation_time = models.TextField(null=True, blank=True)
    components = json.JSONField(default={}, null=False)

    class Meta:
        db_table = 'embedded_sensornets'
```

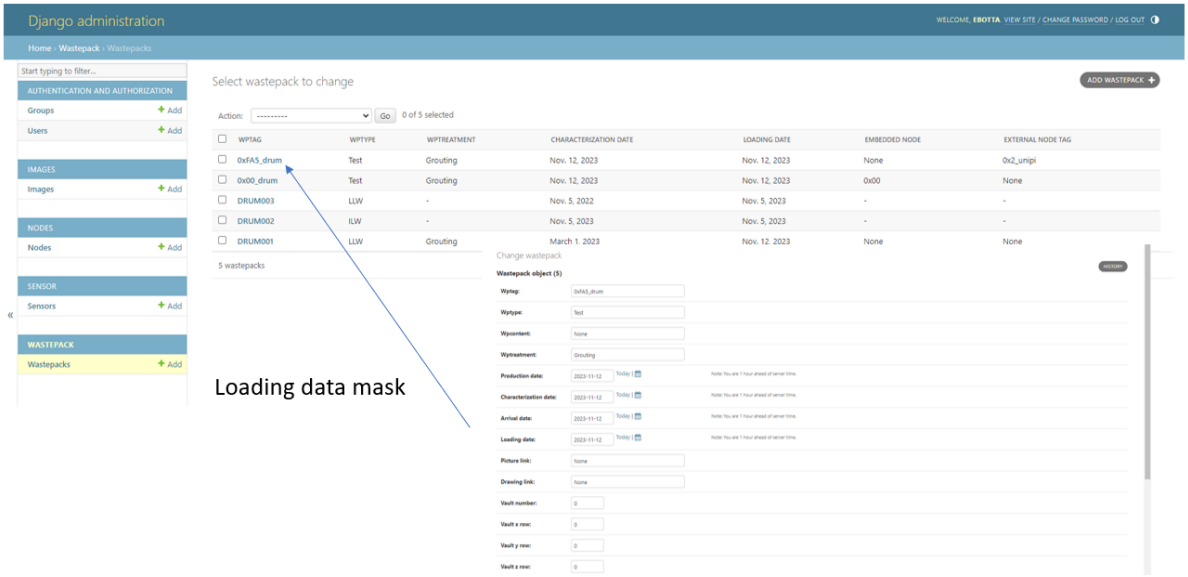
Django can be run in administration mode:



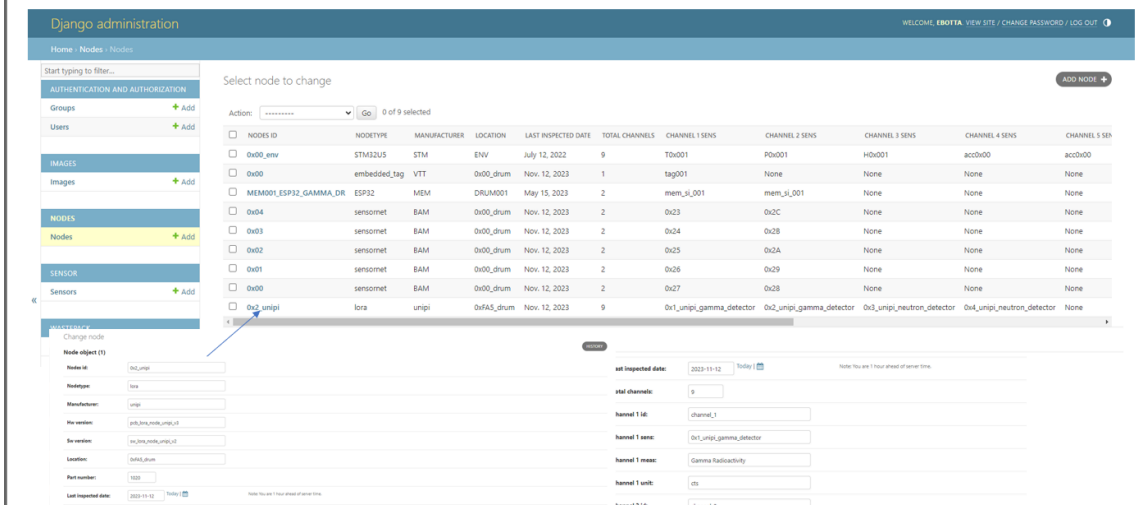
Django can be run in viewer mode:



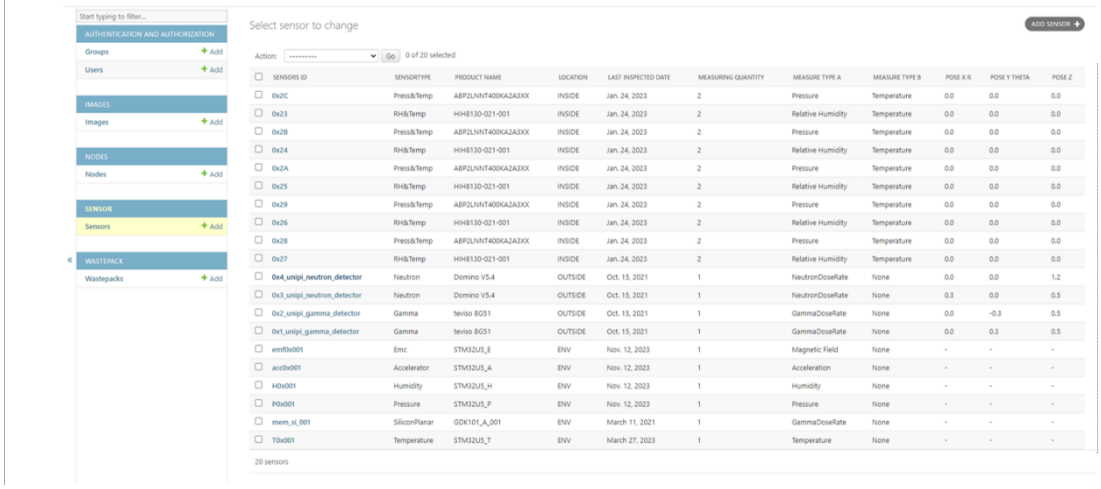
WP 7.5.1 Django – Admin mode - Wastepacks windows



WP 7.5.1 Django – Admin mode - Nodes windows



WP 7.5.1 Django – Admin mode - Sensors windows



InfluxDB as NoSQL reference database

To manage time series the reference tool is influxDB (www.influxDB.com)
Three different implementations of InfluxDB are available:

1. OSS.X (Client and Server are on your local computer or on your VM in the Cloud)
2. On Cloud: Buckets are stored on InfluxDB rent server WWW (Azure in UE, AWS in UK and USA, …)
3. Enterprise for Professional Services (Many Organization Interacting) on Cloud (fee on project size – no limits in bucket size)

Different InfluxDB versions are available on web to be downloaded, with difference performances and limits.
InfluxData Inc' Objective (USA Company – Headquarter in San Francisco, CA) is to do simple and «cheap» what it is costly on MS AZURE and AWS for the services related to the VM and networking services.
In Europe, InfluxDB on Cloud is accessible through (under MS Azure Platform in Netherland) https://europe-west1-1.gcp.cloud2.influxdata.com/orgs/aa8059aa9f79edb5
User can register (for free) by Google or Microsoft personal account and have one Bucket (i.e., database) to start a personal project with drastic limitation on the total number of bucket and data retention (limited to 30 days) as well as API and integrated dashboards.
User can also register for a local Client version OSS.X downloaded on his PC.
In PREDIS Task 7.5 project, VTT had defined a VM on MS Azure with an OSS.X version installed (version 2.3) at http://20.160.36.135:8086/

Time series data management requirements are:

- Data are immutable
- Writing in append
- Reading contiguous sequence of samples data.
- Highly compressible data.
- Deleting usually across large time period
- High precision for short period of time
- Single value is not so important (i.e., not critical)

In order to manage them, two roads are possible:

a) Use a pseudo commercial tool as InfluxDB, which allows 4 Steps in One for Data Management (Acquire, Enrich, Operate, Analyze)
b) Combine a set of services:
    a. NotOnlySQL DB (i.e., MongoDB, SQLite3)
    b. Graphical Tools as Graphana, CSS Studio
    c. Additional services on Clouds (Azure, Google, AWS) to run scripts to transfer data.

When a project is set-up, the user shall consider:

- In OSS.X InfluxDB (the version installed by VTT on MS Azure VM), all buckets (i.e., databases related to the project) shall belong to the same "Organization"
- In Cloud InfluxDB: User pays a fee for the number of buckets used (4 payloads); in the Enterprise option there are not limits: in any case 1 or 2 buckets are free to set-up the case. Access to the buckets by different members is governed by the administrator of the main organization, which had open the account.

Acquire time series data from Subtask 7.5.3 is not different from acquire data in the frame of Industry 4.0 IoT telemetry since any event is characterized by a timestamp.

Analysis of IoT application shows two architectural patterns as dominant:

- Hub only
- Edge and hub

Hub-only IoT data architectures are where each of “things” send their IoT telemetry to a central time series database (such as InfluxDB Cloud version) for storage, enrichment and analysis.
All devices are assumed to have a reliable, performant Internet connection to send their IoT telemetry, and thus all consumers of that data access it from that centralized database.

Edge and hub data architectures are required when a piece of equipment doesn’t have a network connection that is fast and reliable, and where people are onsite and require access to the analytics to assist in operational decision making, while centralized visibility across all of the potential edge sites can be leveraged to understand trend analysis, operational efficiencies or identify potential issues.
It makes sense to dual-write time series telemetry to an onsite time series database, as well as to a centralized instance.
The edge-and-hub pattern is a InfluxDB schema to fulfil:

- Take measures on field with a local hub
- Send selected data remotely (also offline is connection does not work) to a common control room
- Perform analysis of predictive maintenance and send back the results

The investigated strategy to acquire data from the field started by defining two different basic buckets:

1) One strictly to the acquire data from the sensor independently from they are
2) One related to the waste-package as described in the SQL database where sensors are associated and measure flow from “sensor” bucket by dynamic queries.

With the same logic, other buckets can be defined (i.e., to collect data from simulation and/or from prediction)

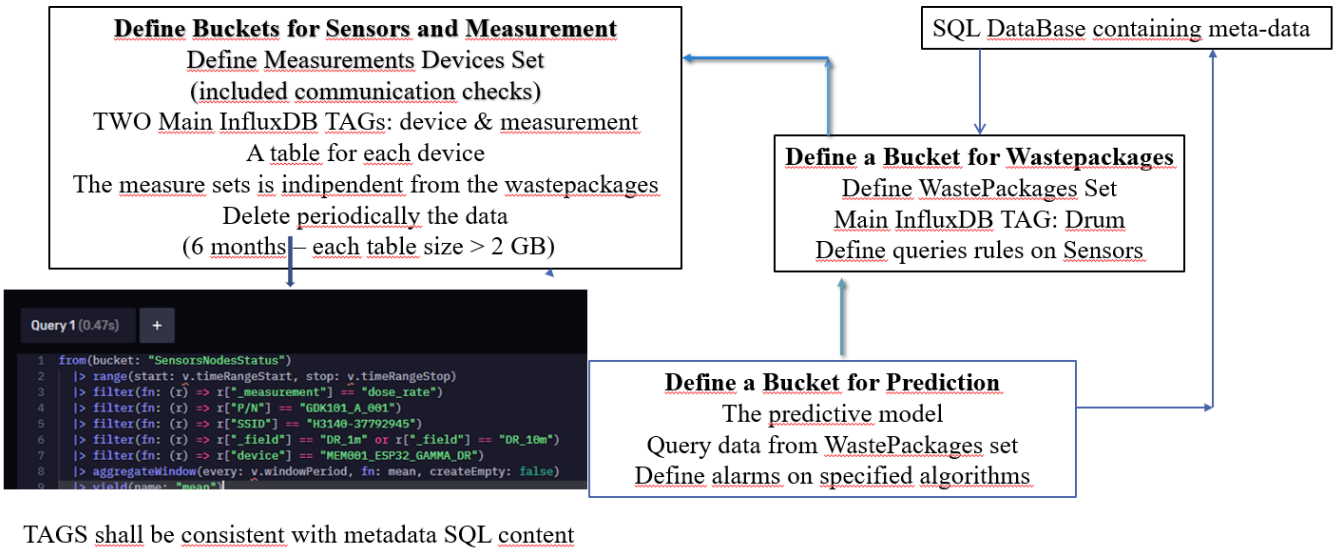


Figure A1.10. InfluxDB Data Flow Management Schema.

InfluxDB can read SQL databases from many sources:

- Amazon RDS: https://aws.amazon.com/
- CockroachDB: https://www.cockroachlabs.com/
- BiqQuery: https://cloud.google.com/bigquery
- MariaDB: https://mariadb.org/
- MySQL: https://www.mysql.com/it/
- Percona: https://www.percona.com/
- PostgreSQL: https://www.postgresql.org/
- SAP HANA: https://www.sap.com/italy/products/technology-platform/hana.html
- Snowflake: https://www.snowflake.com/it/
- SQLite: https://www.sqlite.org/index.html
- Vertica: https://www.vertica.com/

In these case studies, interface is limited to SQLite 3 and PostgreSQL.

Modules to perform I/O data management (API)

Acquiring data in influxDB buckets can be performed by Application Programming Interface (API). Two options are possible:

1. By a Data Acquisition Streaming or Pipeline:
    a. Telegraf Agent (Telegraf service shall run on your PC at boot)
    b. Arduino Client Libraries
    c. Native Integrations (Apache NI)
2. Off Line Data Acquisition, which implies
    a. CSV format with headers describing the structure of file compliant with InfluxDB requirements: a “Field” and a “TimeStamp” shall be always present.
    b. Use On line protocol Commands (strongly suggested design choice)

Telegraph agent includes many industrial plugins (OPC UA, Modbus, KNX, ExecD, MQTT, …). The more efficient way to proceed is to implement directly the communication inside the DAU (or sensor node) Firmware. A native set of instructions are proposed on web:

- OPCUA: https://www.influxdata.com/integration/opcua/
- MODBUS: https://www.influxdata.com/blog/how-to-monitor-your-modbus-devices-with-influxdb/
- KNX: https://github.com/influxdata/telegraf/tree/1eb47e245c0c22270aaf9a42938f5f5f6697a959/plugins/inputs/knx_listener (not validated)
- MQTT: https://www.influxdata.com/mqtt/
- AMQP:https://www.influxdata.com/integration/amqp/
- AZURE Event HUB: https://github.com/influxdata/telegraf/blob/release-1.18/plugins/inputs/eventhub_consumer/README.md (not validated)
- RTI DDS: https://www.rti.com/developers/rti-labs/telegraf-plugin-for-connext (external organization for fee)
- Native Support Apache NiFi, openHAB, WinCC, Node-RED, Particle.io, Arduino (including ESP32, LoRA)

Testing the Real Time Acquisition Schema

A preliminary set-up has been defined on influxDB VM by creating two buckets: SensorNode and WastePackage

In order to generate a real time data pipeline with influxDB, a silicon detector have been connected to an ESP32 device configured to exchange data with a router by WiFi in order to be routed on Cloud by respecting the security rules required by Microsoft on Azure and/or influxDB organization (which in Europe is using MS Azure as reference cloud platform).
In particular, ESP32 WROOM 32 is configurated as a Station and the Access Point is the router:

The basic information inserted in the firmware by Arduino (version 2.0) to connect to MS Azure are:

```
Gamma Silicon Sensor's Interface
 *   Combination with the built-in Wire library to interface.
 *   ESP32 analog input 22 - I2C SCL
 *   Esp32 analog input 21 - I2C SDA

#include <WiFi.h>
// #include <WiFiMulti.h> / need if device is an AP
// be aware esp_arduino_version.h need to be copied manually in influxdb library
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h> // only to manage cloud certificates

#define DEVICE "MEM001_ESP32_GAMMA_DR"
// sensors connected to the device
#define PNUMBER "GDK101_A_001"

// Router where ESP32 Station shall be connected
// WiFi AP SSID
#define WIFI_SSID "H3140-37792945"
// WiFi AP password
#define WIFI_PASSWORD xxxxxxxxxx

The above strings are needed to check the secure connection on Cloud side.
// InfluxDB Cloud services data for RT connection
// InfluxDB v2 server url, e.g. https://eu-central-1-1.aws.cloud2.influxdata.com (Use: InfluxDB UI -> Load Data -> Client Libraries)
#define INFLUXDB_URL "http://20.160.36.135:8086"
// InfluxDB v2 server or cloud API token (Use: InfluxDB UI -> Data -> API Tokens -> <select token>)
#define INFLUXDB_TOKEN "ySjcsTDpAOavkgUwPVVrouUfeljTaYZyjq509gpp0Nq_b9AgZlYNfTakOXwW8u3-_kMIP7hGm5RAf1biCvjKVA=="
// InfluxDB v2 organization id (Use: InfluxDB UI -> User -> About -> Common Ids )
#define INFLUXDB_ORG "VTT"
// InfluxDB v2 bucket name (Use: InfluxDB UI ->  Data -> Buckets)
#define INFLUXDB_BUCKET "SensorsNodesStatus«

……follows instructions to define parameters of your sensor-nodes....

Specific token shall be generated for each bucket to open connection.
If https option is used, security option shall be disabled or remote programmed.

// Set timezone string according to https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html
//  Central Europe: "CET-1CEST,M3.5.0,M10.5.0/3"
#define TZ_INFO "CET-1CEST,M3.5.0,M10.5.0/3" // InfluxDB mandatory
```

```
// InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN,
InfluxDbCloud2CACert);
// Data point to be transferred to InfluxDB (definition of measurements inside the bucket)
Point sensor("dose_rate");

In void setup ()
//Add tags for iInfluxDB Bucket
  sensor.addTag("device", DEVICE);
  sensor.addTag("SSID", WiFi.SSID());
  sensor.addTag("P/N", PNUMBER);
  // Accurate time is necessary for certificate validation and writing in batches
  // For the fastest time sync find NTP servers in your area: https://www.pool.ntp.org/zone/
  // Syncing progress and the time will be printed to Serial.
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");

….. follows instruction to set-up your sensor node.....

// Check connection with server where InfluxDB is installed
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }
…

In void loop ()
...write first the instructions to get the measure (in this case average DR last minute and over 10 minutes) .....

// Clear fields for reusing the point. Tags will remain untouched
  sensor.clearFields();
  sensor.addField("DR_1m", value1m);
  sensor.addField("DR_10m", value10m);
  // Store RSSI value into point (it indicates local connection status)
  // Report RSSI of currently connected network
  // RSSI is considered a Field (i.e. data to be measured)
  sensor.addField("rssi", WiFi.RSSI());
  // Print what are we exactly writing
  Serial.print("Writing: ");
  Serial.println(sensor.toLineProtocol());

…put instruction to clean local memory of your sensor-node or restart the loop...
```

The above strings define what is transferred in the bucket SensorNode by including the effective connection check to prevent from hacker attack.

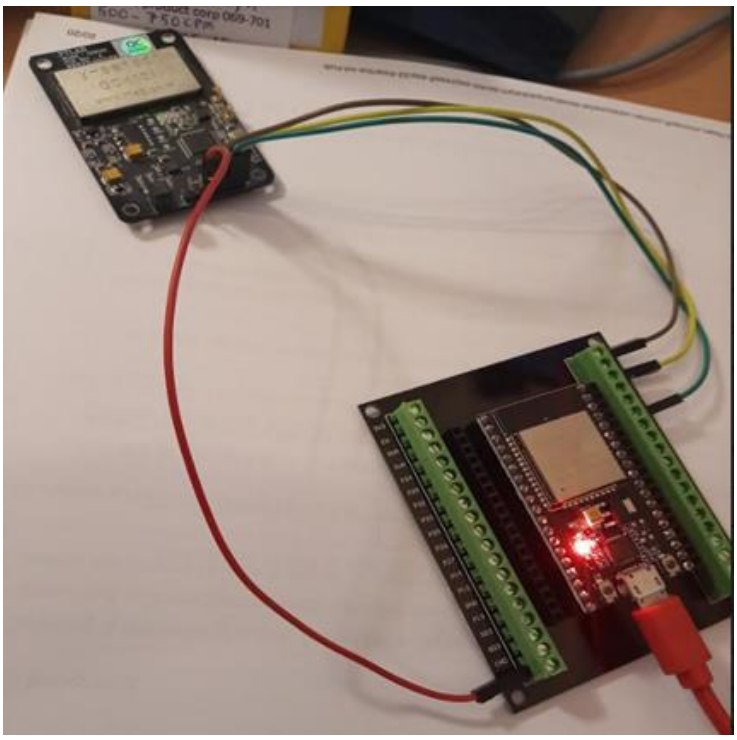


Figure A1.11. Silicon detector plus ESP32 node.

Figure A1.12. Real Time Data Acquisition by InfluxDB data explorer.

Data can also be retrieved and transferred to influxDB from other IoT platform or sources (i.e., AWS and MS Azure). In particular:

- InfluxDB OSS.X can acquire data from AWS IoT applications only if data are previously saved in a PostgreSQL Database
- InfluxDB OSS.X can acquire data stored on MS Azure in two different ways:
    - If data are saved in “DataStorage Blob Container” only data are in .CSV format: JSON files needed to be converted by using dedicated services or downloaded by AZCopy in PC/VM by the Storage Account Administrator Only

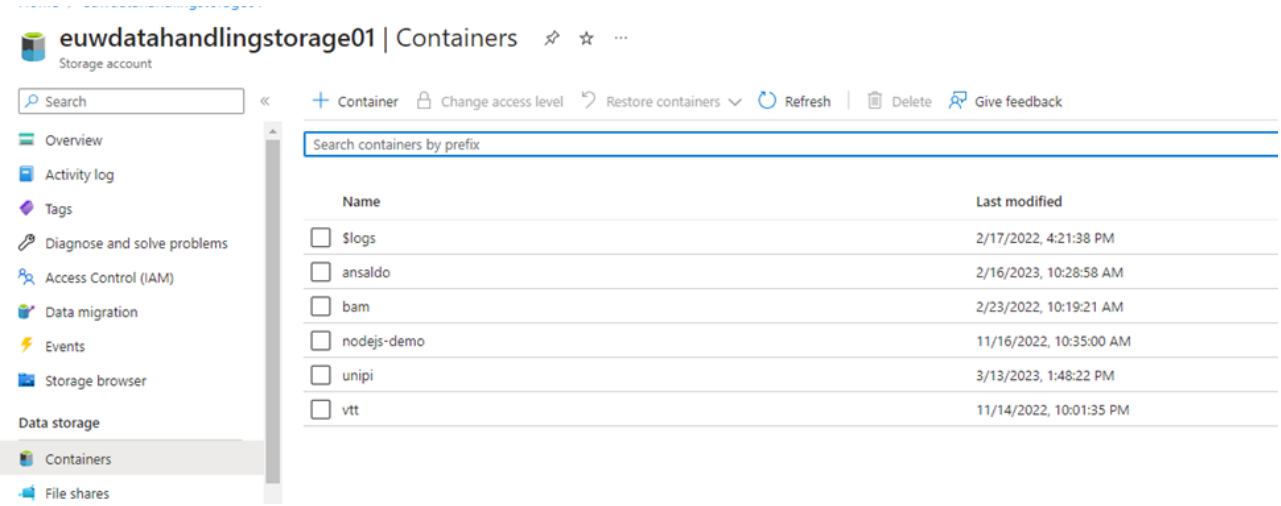


Figure A1.13. Screenshot of PREDIS project containers in MS Azure.

- If data are real time data saved on IoT Central, it is required that data are exported and saved on MS Azure Event HUB able to ingest large quantities of data (https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-about). This MS service is oriented and sized to acquire Big Data

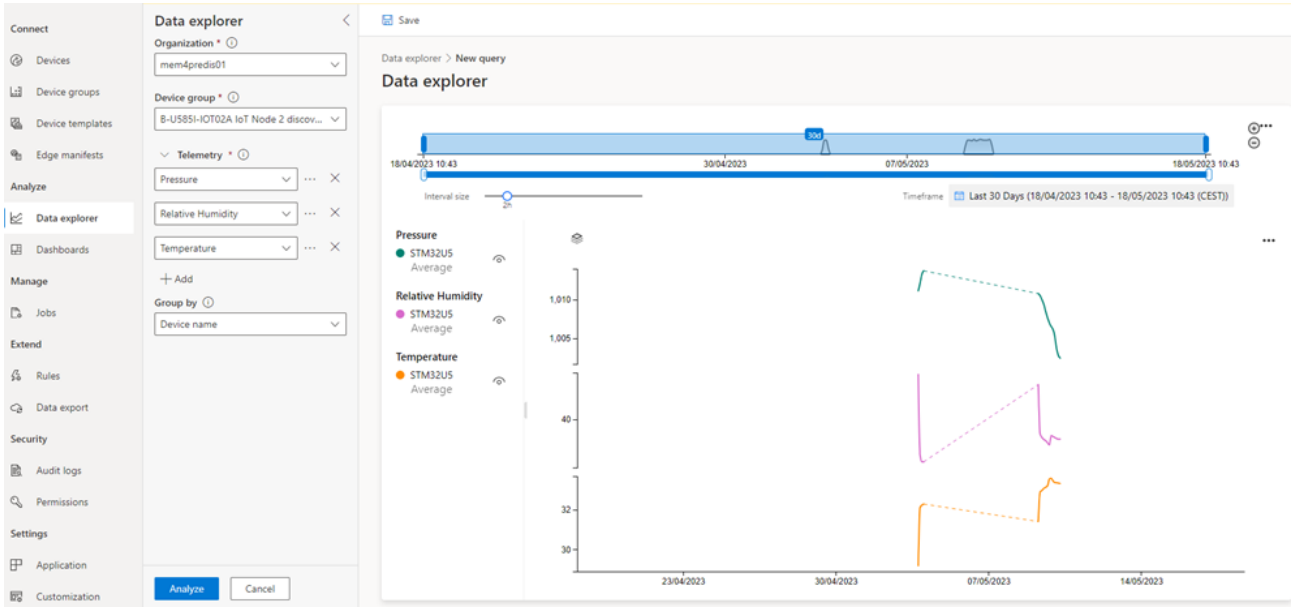
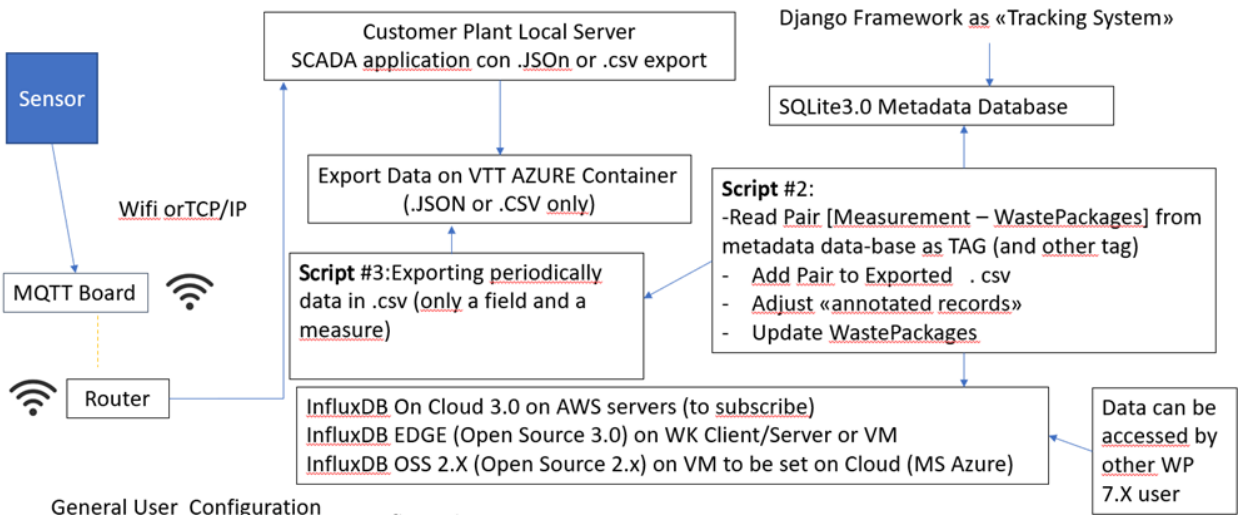


Figure A1.14. Screenshot of Data Explorer service on MS IoT Azure Central.

Potential Communication Schema in Data Handling

A general data communication schema has been setup by dedicated scripts.



•Miniconda 3 is the reference VM where create inside a Python 3.11 environment

•Working Test: Windows 10 / 11 Professional on Local WK

•Python virtual environment (venv) shall be set for any application (i.e., one for pandas (influxdb 3.0), one for pandas (influxdb 2.x), one for Django (Sqlite 3.0 database) to avoid library conflict)

•The following module shall be installed for data management (up to now):

•InfluxDB 3.0 or EDGE (using SQL): “pandas”, “datafusion”, “pyarrow”, “parquet”, “tabulate”, “matplotlib”, “influxdb3-python”

•InfluxDB 2.x OSS (using Flux): “pandas”, “datafusion”, “tabulate”, “matplotlib”, “influxdb-client-python

Two buckets (real time databases) have been defined in InfluxDB: “SensorsNodeStatus” which acquire measure and instrument status from field and “WastePackage” where sensors and measure are associated to the concrete drum in real time by reading part of the information from the SQLite3 database (association between sensors and drums) and from the “SensorsNodeStatus”

Procedure:

1.Access to «SensorsNodeStatus» database by using «token» released by organization A

2.Perform two queries on a selected time frame:

•One as table to see the raw data and influxdb schema (for debug)

•One oriented to create directly a pandas dataframe (to manipulate)

3.Check the pandas data frame by eliminating dirty raw («NaN») in cell to transfer

4.Drop the columns from pandas data frame if they are not to be transferred to «WastePackage» influxdb database (i.e. communication line data as SSD, rssi)

5.Check by plotting the data are you going to transfer (optional by matplotlib functions)

6.Read the «metadata» associated to «device» in SQLite3 database (managed by Django) to use as TAG in «WastePackage» (script is on going)

7.Add as columns (on the right) in pandas dataframe addressed to contain the TAGs read at point 6

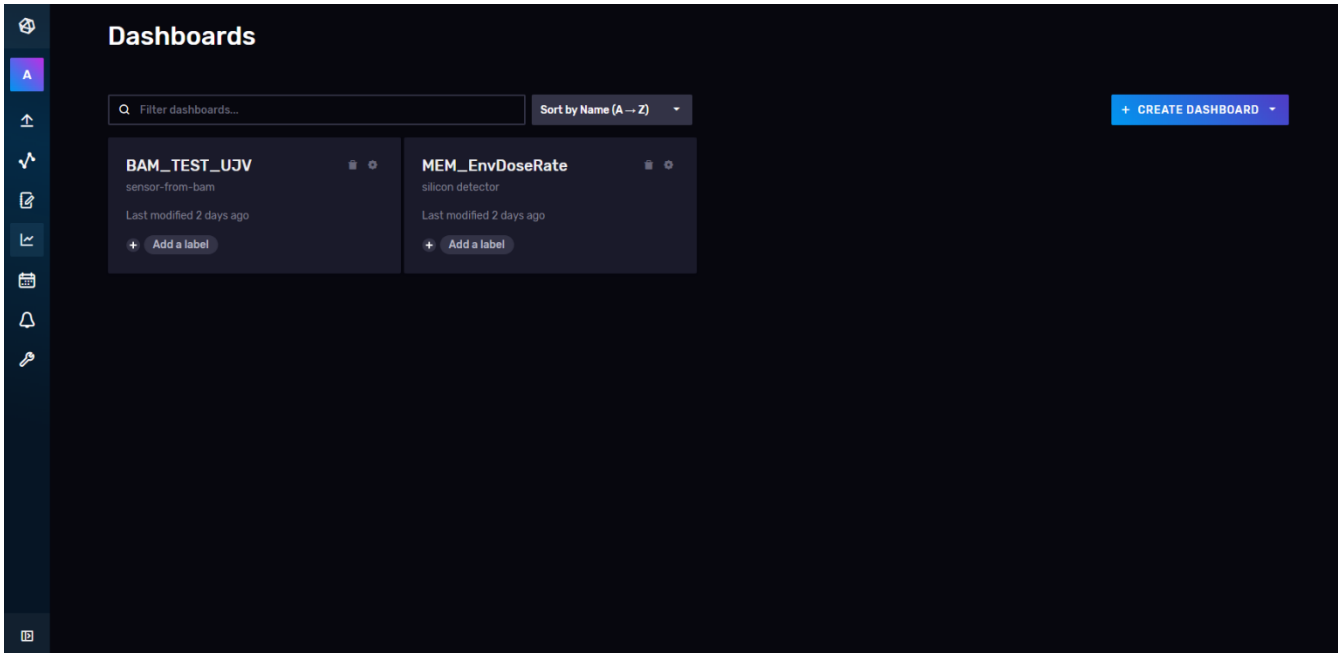
8.Check new pandas dataframe consistency (optional)

9.Convert the pandas dataframe series in Apache Arrow (use pyarrow)

10.Save the produced arrow as a .parquet file (read in input to influxdb)

11.Access to influxdb «WastePackage» database by using «token» released by organization B (data user cannot belong to A or B)

12.Upload the data in influxdb bucket (if there is not error message: data will be available on cloud after about 60 s for data exploring from other authorized organization).



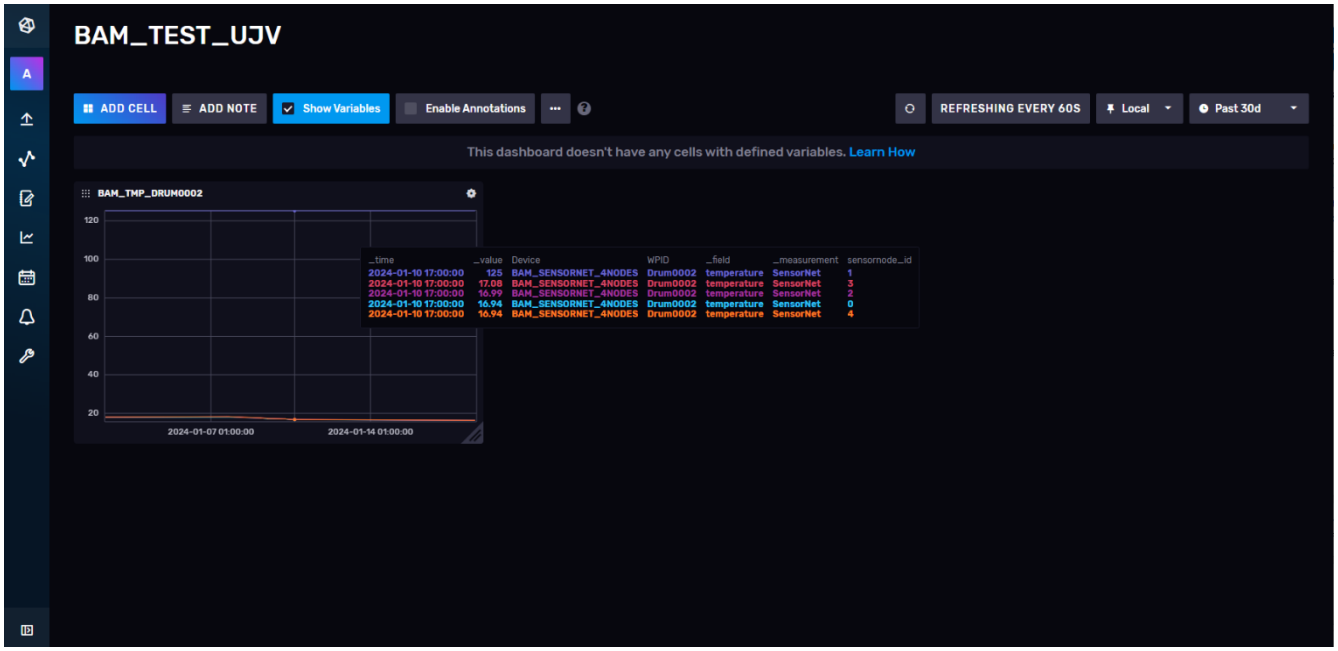


Figure A1.15. Alternative dashboard based on InfluxDB End User Customable.