WP4 – Digitalization of operational processes, smart sensors configuration and DSS implementation

T 4.2 – Sensors networking and data fusion mechanisms

Person responsible / Author:	Dimitris Ntalaperas (SINGULAR)
Deliverable No.:	4.2
Work Package No.:	WP4
Date:	31.10.2019
Project No.:	760792
Classification:	Public
File name:	FENIX_D4.2_Sensors networking and data fusion mechanisms.docx
Number of pages:	22

Status of deliverable

Action	By	Date (dd.mm.yyyy)
Submitted (author(s))	Dimitris Ntalaperas	31.10.2019
Responsible (WP Leader)	Dimitris Ntalaperas	
Reviewed by	Aristotelis Spiliotis, Paolo Rosa, Roberto Rocca	31.10.2019

Revision History

Date (dd.mm.yyyy)	Revision version	Author	Comments
20.06.2019	v0.5	Dimitris Ntalaperas	Draft version
28.06.2019	v1.0	Dimitris Ntalaperas	Final version
24.10.2019	v0.9	Dimitris Ntalaperas	Draft Revised Version
30.10.2019	v1.0	Dimitris Ntalaperas	Final Revised Version

Author(s) contact information

Name	Organisation	E-mail	Tel
Dimitris Ntalaperas	SINGULAR	dimitris.ntalaperas@gmail.com	+30 6944029042

H2020 Innovation Action - This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N. 760792

ABSTRACT

The current document reports the development of the mechanism responsible for collecting and storing stream and legacy data, originating from sensors or other external devices as well as, relational databases or files containing structured data. The mechanism supports both real time and off-line collection provided that the on-site infrastructure retains the data or uses an intermediate storage service. The infrastructure developed provides a set of adapters that can connect to a variety of sensors. Data retrieved from sensors are then filtered and refined online, by using a Storm Infrastructure, and are uplifted to the DSS Semantic Model. They are then stored to a times series database which can be queried both by the users to extract meaningful information and by the DSS to analyse patterns or identify outliers that can generate alerts. Static data are likewise uplifted according to the semantics of the DSS model and are stored in suitable database for querying and analysing.

H2020 Innovation Action - This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N. 760792

Table of Contents

Figures

Tables

Abbreviations and Acronyms:	
CBM	Circular Business Model
DSS	Decision Support System
OWL	Ontology Web Language
PCB	Printed Circuit Board
PLC	Programmable Logic Controller
RDF	Resource Description Framework
XML	eXtensible Markup Language

INTRODUCTION

Deliverable 4.2: Sensors networking and data fusion mechanisms documents the infrastructure developed which is responsible for collecting data from sensors installed in the industrial plant, refining them, uplifting them to be aligned with the DSS Semantic Model and finally stores them in a suitably configured time series database. The document is structured as following:

- Section 1 presents the architecture of the module that implements the data fusion mechanism
- Section 2 describes how the low-level sensor data collection mechanisms are implemented together with the filtering mechanism that allows data normalization and refinement at real time
- Section 3 documents how data are aligned with the semantic model and are pushed into the time series database either directly or through an event broker.
- Section 4 gives an overview of how data are typically structured and queried once they are stored in the time series database
- Section 5 finally, gives the main conclusions of the present work.

1. REQUIREMENTS ENGINEERING

The current section analyses the business requirements of WP1, shows how they are mapped to technical requirements and, how the last are reflected to the current architecture of the Sensors Networking and Data Fusion Mechanisms. The core functional requirements as well as the non-functional requirements, and the outcomes of D4.1, have been further elaborated and defined a new set of technical requirements addressed by the Sensors networking and data fusion mechanisms infrastructure. Finally, each technical requirement is mapped to the relevant subcomponent of the architecture of the Sensors Networking and Data Fusion Mechanisms.

1.1. Functional Requirements

Table 1 presents the core functional requirements of the FENIX DSS Platform, as they are derived from the analysis of the WP1 outcomes, as well as, T4.1. The unique ID of the requirement has been provided for each functional requirement, a brief description of what this requirement entails, the component that should address the requirement and, finally, the source from which this requirement originates (a use case or the results of a Task).

It should be noted that this process is iterative and that some requirements may be refined as the DSS is applied and evaluated by the pilots. Furthermore, a requirement may be fulfilled by more than one component; in this case it is identified accordingly in the "Component" column.

Table 1: Functional Requirements

ID	Need	Description	Component	Input/Use case
FR1	Stream data processing	FENIX DSS Platform should be able to process stream data	DSS Data Fusion	D1.2
FR2	Legacy data processing	FENIX DSS Platform should be able to process legacy data	DSS Data Fusion	D1.2
FR3	Processing data from	FENIX DSS Platform should be able to process data from web sources	DSS Data	UC2

	web sources	(social web platforms)	Fusion	
FR4	Energy data processing	FENIX DSS Platform should be able to define, access and process energy data	DSS Data Fusion	D1.2
FR5	Thermal data processing	FENIX DSS Platform should be able to define, access and process thermal data	DSS Data Fusion	D1.2
FR6	Materials data processing	FENIX DSS Platform should be able to define, access and process material data	DSS Data Fusion	D1.2, UC3
FR7	Resources data processing	FENIX DSS Platform should be able to define, access and process resources data	DSS Data Fusion	D1.2
FR8	Estimation of energy consumption	FENIX DSS Platform should assist in the estimation of energy consumption	DSS Data Fusion, DSS Engine	D1.2
FR9	Estimation of recyclability index	FENIX DSS Platform should assist in the estimation of the material recyclability	DSS Analytics, DSS Engine	D1.2 (recyclability index)
FR10	Estimation of circularity indicators	FENIX DSS Platform should assist in the estimation of the circularity indicators	DSS Analytics, DSS Engine	D1.2 (circularity indicators)
FR11	Product Requirements definition	FENIX DSS Platform should provide the ability to define the product requirements from customer's side	DSS Engine	UC1
FR12	Correlation models definition	FENIX DSS Platform should provide the ability to define rules and correlations between the recycled raw material characteristics to final dowser properties	DSS Engine	UC1
FR13	Real time user validation support	FENIX DSS Platform should provide the ability to define rules based on user validation and assessment input of the business models	DSS Engine	UC3
FR14	Health and safety assessment	FENIX DSS Platform should provide the ability to support decision support for health and safety assessment	DSS Analytics	UC3
FR15	Provision of performance-based models and metrics about regarding efficiency and sustainability	FENIX DSS Platform should provide performance-based models and metrics about regarding efficiency and sustainability	DSS Analytics	WP1, all use cases

FR16	Provision of patterns recognition techniques	FENIX DSS Platform should provide patterns recognition techniques for recognizing specific trends	DSS Analytics	WP1, all use cases
FR17	Provision of algorithms for statistical analysis, trends and forecasting analysis, classification algorithms etc.	FENIX DSS Platform should provide algorithms for statistical analysis, trends and forecasting analysis, classification algorithms etc.	DSS Analytics	WP1, all use cases
FR18	Alerting and signaling mechanisms for patterns analysis and outliers identification	FENIX DSS Platform should provide alerting and signaling mechanisms for patterns analysis and outliers identification	DSS Analytics, DSS Engine	WP1, all use cases

1.2. Non-Functional Requirements

To identify the not functional requirements, the model proposed by ISO/IEC 25010:2011 has been adopted. There are eight quality characteristics contributing to software product quality according to the above ISO model (Figure 1).

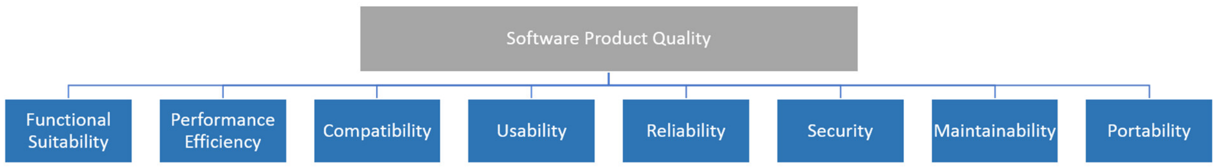


Figure 1: ISO/IEC 25010:2011 Software Product Quality model

Each characteristic deals with a specific framework within the software product quality;, For better understanding, they have been detailed (ISO/IEC 25010:2011) in sub-categories as seen in Table 2.

Table 2: ISO/IEC 25010:2011 Software Product Quality Model Sub-Categories

Quality characteristic	Sub-categories
Functional Suitability	• Functional Completeness • Functional Correctness • Functional Appropriateness
Performance Efficiency	• Time Behaviour • Resource Utilization • Capacity
Compatibility	• Co-existence • Interoperability
Usability	• Appropriateness Recognisability • Learnability • Operability • User Error Protection • User Interface Aesthetics • Accessibility
Reliability	• Maturity • Availability • Fault Tolerance • Recoverability
Security	• Confidentiality • Integrity • Non-repudiation • Authenticity • Accountability
Maintainability	• Modularity • Reusability • Analysability • Modifiability • Testability
Portability:	• Adaptability • Installability • Replaceability

Table 3**Fehler! Verweisquelle konnte nicht gefunden werden.** presents the non-functional requirements of the FENIX DSS Platform taking into account the eight aforementioned qualities. As the non-functional requirements should describe the general operation of the system, there is no assignment of specific component to each one of them; although some NFRs may be more relevant to some components than to others, the platform, as a system, should fulfill all NFRs.

Table 3: Non-functional Requirements (NFR)

ID	Requirement Sub-category	Description
NFR1	Functional Suitability	FENIX DSS Platform should be able to perform a great variety of analysis on selected multiple datasets (private or public) in order to provide state of the art analytics
NFR2	Performance efficiency	FENIX DSS Platform should be able to perform analytics in a timely and efficient manner
NFR3	Performance efficiency	FENIX DSS Platform should be able to guarantee the full optimization of the response time to ensure a functional and flexible navigation
NFR4	Compatibility	FENIX DSS Platform should be able to interact and exchange information with other products / IoT devices in a secure way (for example secure REST API)
NFR5	Usability	FENIX DSS Platform shall feature a user-friendly interface, provide an overview of supported kind of analytics and visualization
NFR6	Reliability	FENIX DSS Platform should ensure high availability of the system and the stored datasets
NFR7	Security	FENIX DSS Platform should offer login with user credentials
NFR8	Portability	FENIX DSS Platform should be able to be deployed in a timely and efficient manner

1.3. Technical Requirements

The technical requirements were driven from the list of functional and non-functional requirements as defined in the previous section and with the mission to utilize the goals and expectations of the users and the stakeholders.

The following list depicted in Table 4 defines the technical requirements that will pave the way for the design and the implementation of Sensors Networking and Data Fusion Mechanisms. Each technical requirement is mapped to the relevant functional and non-functional requirements, as well as, to one or more architectural sub-components of the Sensors Networking and Data Fusion Mechanisms to assure that all technical requirements are covered by the component architecture (Figure 1).

Table 4: Technical Requirements

ID	Functional, Non-functional	Need	Description	Sub-component	Users
TR1	FR1, FR4, FR5, FR6, FR7	Stream data processing	FENIX DSS Platform should be able to process stream data	Time Series DB	D1.2
TR2	FR1, FR4, FR5, FR6,	Sensor data	FENIX DSS Platform should be able to provide	Data refinement, IoT	D1.2

	FR7	acquisition	connectors to sensors	gateway	
TR3	FR1, FR6, FR7	PLC data acquisition	FENIX DSS Platform should be able to provide connectors to PLCs	Data refinement, IoT gateway	D1.2
TR4	FR1, FR6, FR7	S3 data acquisition	FENIX DSS Platform should be able to provide connectors to S3 buckets	Data refinement, IoT gateway	D1.2
TR5	FR2, FR1, FR4, FR5, FR6, FR7	Relational data processing	FENIX DSS Platform should be able to connect to legacy databases and process relational data	Legacy data uplifting, Static storage	D1.2
TR6	FR2, FR3, FR1, FR4, FR5, FR6, FR7	Structured data processing	FENIX DSS Platform should be able to import and process structured data (csv, excel, JSON, XML)	Legacy data uplifting, Static storage	D1.2, UC2, UC3
TR7	FR1-FR8, FR14	Mapping of all data to a common reference model	FENIX DSS Platform should be able to map all imported data to a common reference model	DSS Semantic Model	WP1, D4.1, all use cases
TR8	FR1-FR8, FR14	Harmonization of data before storage	FENIX DSS Platform should be able to apply lightweight harmonization processing to data before storage in local database	Sensor data uplifting, Legacy data uplifting	WP1, D4.1, all use cases
TR9	FR2, FR3, FR1, FR4, FR5, FR6, FR7	Linked data acquisition	FENIX DSS Platform should be able to access and import Linked Data	Linked data connectors	WP1, all use cases
TR10	NF3, NFR6, FR18	Message routing	DSS Data Fusion should provide a routing and signaling mechanism to the rest of the upper architectural layers	Event Broker	WP1, all use cases

2. CONCEPTUAL ARCHITECTURE OF THE FENIX DSS PLATFORM

Figure 1 depicts the Conceptual Architecture of the DSS Platform. In summary, the architecture consists of four layers; the persistency layers for storage, the middle layer that provides interfaces and core services between storage and higher-level functionalities, the enterprise layer which implements and exposes the main services implementing core business logic and, finally, the presentation layer that integrates all services and provides graphic elements for end users.

More specifically:

- The Persistency Layer consists of all the necessary infrastructure and software needed to store both static and streaming data; MongoDB is used for static data whereas InfluxDB is used for streaming data. A triple store is used to store the Ontology of the DSS Semantic Model as well as any Linked data that will be produced by the DSS.
- The Middle Layer contains all the adapters for interfacing with data and sensors as well as with the ontology. It also contains the uplifting mechanisms that allow the alignment of data with the DSS Semantic Model. Various technologies have been incorporated for dealing with adapters for heterogenous data. ThingsBoard was used for sensor adapters while for static storage Java MongoDB drivers were used. Finally, Apache Jena, was used for interfacing with the triple store.
- The Enterprise Layer contains all core services needed for the DSS to operate efficiently; these contain forecast services, event/alarm signalling services, the patterns recognition services and the Rule Engine of the DSS. For the core business logic of these services, appropriate machine learning libraries of Python will be used (Pandas[1], NumPy[2], Scikit-Learn[3] and Keras[4])
- Lastly, the presentation layer contains all the necessary graphical elements, as well as the integrated graphical platform that facilitate users to use the DSS. The presentation layer will be implemented using Spring Boot framework[5] and Thymeleaf[6] rendering engine. JQuery[7] will be used for front-end programming.

Concerning the interfaces between the layers, these will be mainly developed following a loose coupling approach by using REST. Each upper tier module (middle layer and above) will implement tis own well-defined set of REST interfaces. The interface between Persistency and Middle Tier also employs an alternative queuing mechanism for forwarding sensor data to the InfluxDB database; this is transparent to the rest of the architecture's modules.

[1] https://pandas.pydata.org/
[2] https://numpy.org/
[3] https://scikit-learn.org/stable/
[4] https://keras.io/
[5] https://spring.io/projects/spring-boot
[6] https://www.thymeleaf.org/
[7] https://jquery.com/

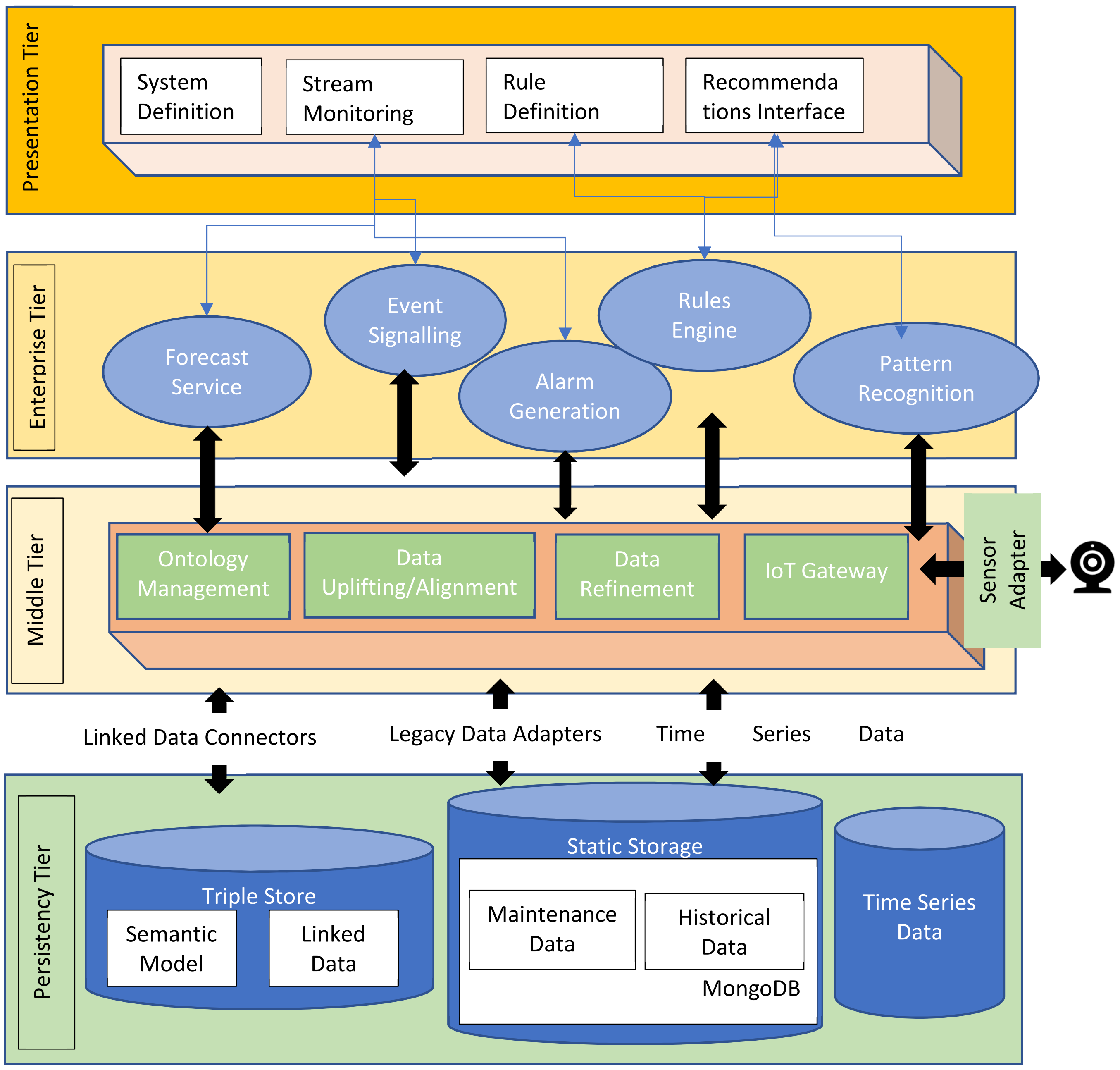


Figure 1: Conceptual Architecture

Not depicted in the diagram is the "Event Broker" component that fulfils TR10. This is one of the two alternative ways to communicate data between Middle and Enterprise Tier and is implied by the connection between these two layers.

Though the various layers of the architecture may overlap regarding the modules for which they provide functionality, the following correspondence between layers may be given:

- Persistency and Middle Tier is implemented mainly by the Sensor networking and data fusion mechanism module that is the focus of the present document

- The services of the Enterprise Tier, bar the Rules Engine, are implemented by the Analytics Services module that is the main result to be produced in Task 4.3; the results will be documented in D4.3
- The Rules Engine is implemented by the DSS Engine module that is the main result to be produced in Task 4.4; the results will be documented in D4.4. The Presentation Layer is also considered part of the DSS Engine and will likewise be reported in D4.4.

3. ARCHITECTURE OF SENSORS NETWORKING AND DATA FUSION MECHANISMS

Figure 2 depicts the technical architecture of the DSS Data Fusion Mechanism. It broad terms, it corresponds to the Middle Tier of the Conceptual Architecture (although, strictly speaking, the various storage solutions are not part of the module, they have been implemented in the context of the work done in T4.2 and are therefore also documented in the present report). At the lower level, data are collected through two mechanisms depending on their nature:

- Stream data are collected through an IoT gateway. The IoT gateway is responsible for connecting to the data source and capturing the data as they are generated. It can connect directly to sensors using the ThingsBoard[8] framework or to PLC using the relevant adapters. Since many manufacturers use S3[9] bitbuckets for storing sensor data, the IoT Gateway also provides S3 clients; this solution can be applied when real time analysis is not required, and sensor data can be analysed in batch.
- Legacy data are collected in a per case manner according to the specification of the source files that the user provides. Depending on the case a mapping file is created, and a legacy data adapter collects the data; currently the source formats supported are excel files, csv files, MySQL source files and access database files.

In both cases, once data are captured from the relevant sources, they need to be uplifted to the DSS Semantic Model (see D4.1: DSS semantic model). This uplifting is carried through the usage of mapping files and ensures that data are stored in a universal way. In this way, they can be queried with a single engine in a way that is transparent to the specific business case. This is very critical to the efficient implementation of the modules of the DSS Analytics Services and the DSS Engine which are implemented in the context of T4.3 and T4.4 respectively. Please refer to Annex B for an example of how uplifting is performed using a sample XML mapping.

From these mappings, the module can understand the exact semantics of a time series measurement. The opposite is also true. If an instance of a measurement has the properties defined in the sample XML, the module can "down-lift" it and discern the names of tags and values of the corresponding time series entry.

Once data uplifting is performed, data can be stored for analysis and queries. In the case of Legacy Data, the data are stored in a static database, which for the needs of FENIX is a MongoDB[10], while in the case of stream data, data are stored in an InfluxDB[11], a time series database appropriate for the storage and querying of large amounts of time series data. In the

[8] https://thingsboard.io/

[9] https://aws.amazon.com/s3/

[10] https://www.mongodb.com/

[11] https://www.influxdata.com/

case of streaming data specifically, two storing mechanisms are supported: a) direct, in which the *Sensor Data Uplifting* module stores the data in the InfluxDB and b) via an event broker, where data are forwarded in an event broker and from there stored in InfluxDB. The event broker is implemented in Apache Kafka; this approach is followed in case that streaming data needs to be accessed by another supporting module.

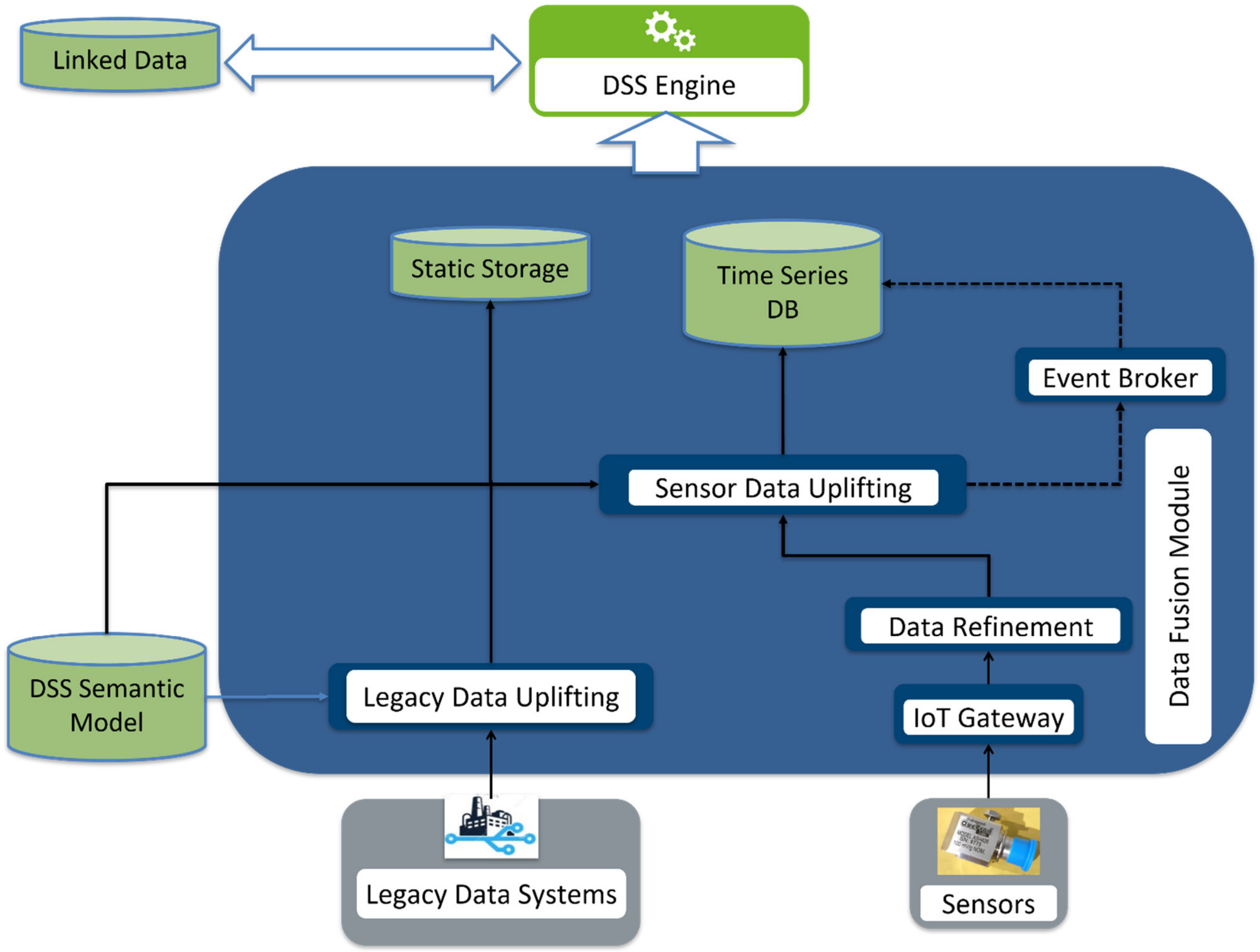


Figure 2: Architecture DSS Data Fusion Mechanism

4. SENSOR DATA COLLECTION

Sensor data collection is carried through the IoT Gateway module which is depicted in Figure 3. Data can be collected directly from sensors using an adapter which is implemented using ThingsBoard IoT platform. If streaming data are to be stored in a PLC a custom PLC adapter is used and is implemented using the S7 connector library. While currently only Siemens PLC are supported, new PLCs can be easily accommodated using Thingsboards. While not technically streaming data, since many manufacturers use Amazon S3 bitbuckets for storing sensor data, a custom adapter that is based on Amazon S3 Client library is also implemented.

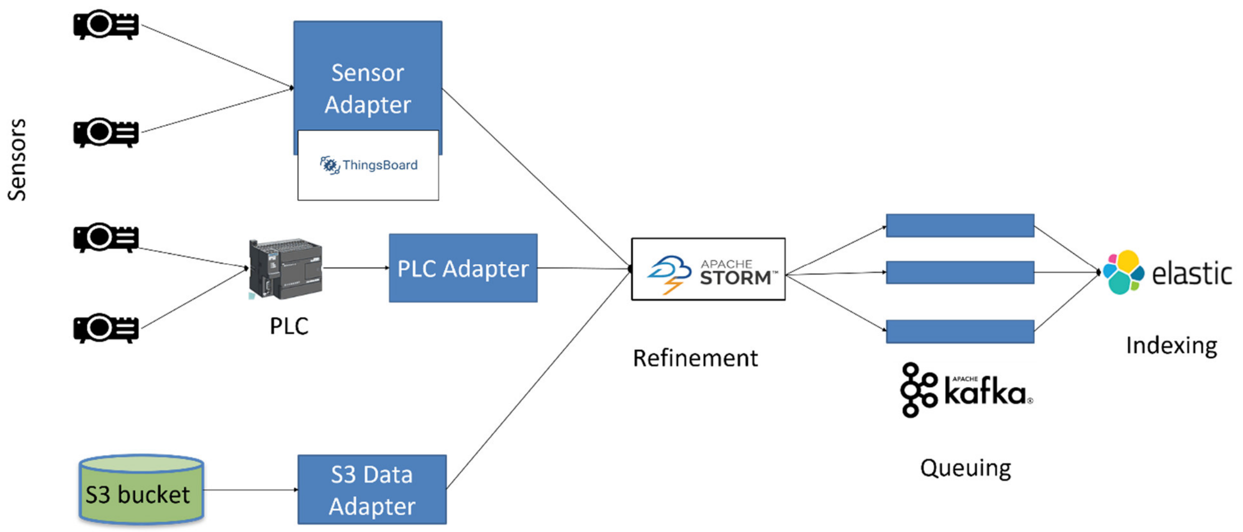


Figure 3: IoT Gateway

Adapters collect data as these are generated from the sources. However, in most cases, these data may need refinement before being stored. While complex operations, like detection of outliers, rests upon the higher-level services that are going to be implemented in the context of DSS Analytics Engine in T4.3, simple operations like rounding up of values, discarding of evidently erroneous values and simple aggregations can be performed upon collection. This is achieved by implementing the required operations using Apache Storm and by implementing specially designed StormBolts which perform the required operations.

Figure 7 (Annex A) depicts a simple Bolt that performs rounding of values up to the second digit. According to the needs of each business scenario, more complex operations can be performed.

4.1. Sensor Data Campaigns in FENIX

In FENIX, POLIMI is the pilot that currently uses sensor data. POLIMI has fully endorsed Industry 4.0 techniques and has established a network of sensors and PLCs that monitors the entire power line. A series of Data Workshops to determine which ones of these measurements are relevant for FENIX has already begun and has formed part of the functional requirements already analysed in Section 1.1; these workshops are to continue in parallel with WP6 activities.

In summary, POLIMI collects data at various phases of the assembly-disassembly process. Workstations are equipped with PLCs that track the production and collect data related both to process control and to energy monitoring. Tracking covers the entire process with data being collected at the stations (e.g. through TwinCat 3 or CodeSys), propagating to the responsible Manufacturing Execution System (MES) and finally being stored statically in an MS Access Database.

Process control data contain, among other, information about each station state with each station being equipped with specialized sensor configuration according to its role. The drilling station for example is equipped with sensors monitoring motion and product inspection, while the robot assembly station is equipped with sensors monitoring robot position, belt movement, PCB working position etc.

Energy monitoring data (e.g. pressure and power consumption) on the other hand contain a wealth of information that can be used for evaluating the performance of the CBMs and potentially give valuable input to the DSS.

Since the data collected in POLIMI's I4.0Lab have a huge volume and real time monitoring by the DSS of all these variables can be both redundant and time consuming, it was decided that analysis will be first performed in the MS Access Data. To this end, the Legacy Data Adapters implemented in T4.2 will be used on the Access database files to align the data to the DSS Semantic Model. Analysis algorithms that are being implemented in T4.3 will then try to identify patterns in the collected data. Depending on the results, the critical variables will be monitored regularly, either through direct sensor adapters in real time or in batch at regular intervals the length of which will be decided in collaboration with POLIMI during the activities of WP6.

For a full specification of POLIMI's I4.0Lab please refer to *D3.1 Simulation of assembly-disassembly processes*

5. DATA ALIGNMENT

Data alignment involves the enrichment of the source data in a way that they are aligned with the concepts of the DSS Semantic Model. By performing alignment, the data can be queried in a universal way that is transparent to the specifics of each variable. Data alignment needs to be performed both in legacy and in streaming data. In the former case a mapping file is required that needs to be parametrized according to each business scenario. Since legacy data can be heterogenous and have highly varying semantics, the schematics of the mapping file will be specialized to each business case. On the other hand, sensor data tend to have more rigid semantics; typically, a variable originating from sensors tends to have a set of values and tags. To this end, the mapping file for sensor data, though also customizable, has a more standard format. This not only allows the production of triples from sensor data but can also be used to have entities and properties from the Semantic Model embedded to the tags and values of the InfluxDB entries.

Figure 4 depicts the above process. Data are uplifted and are enriched with the appropriate metadata of the DSS Semantic Model. These uplifted data are then stored in their respective databases, an InfluxDB for sensor data and a MongoDB for static legacy data. Optionally, the same entries can be converted to RDF triples according to the DSS Semantic Model specification and stored in a triple store. As the DSS Engine is being implemented in the context of T4.4, this optional mechanism may be used for publishing the results of the DSS Engine in a linked data format.

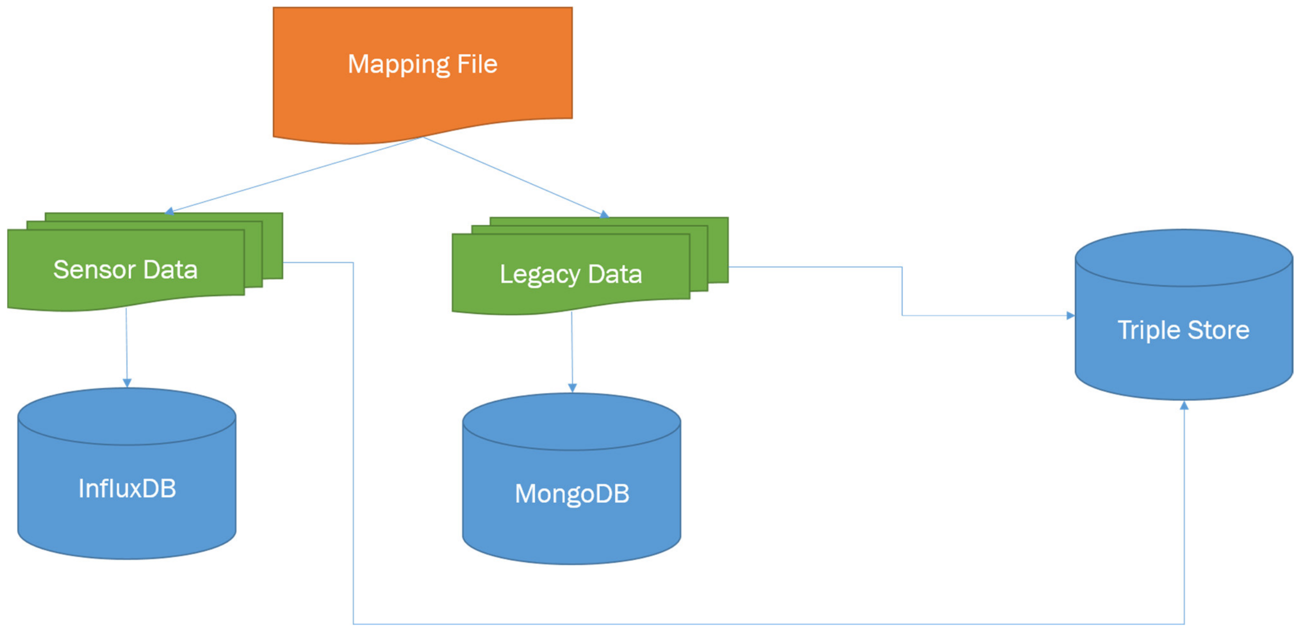


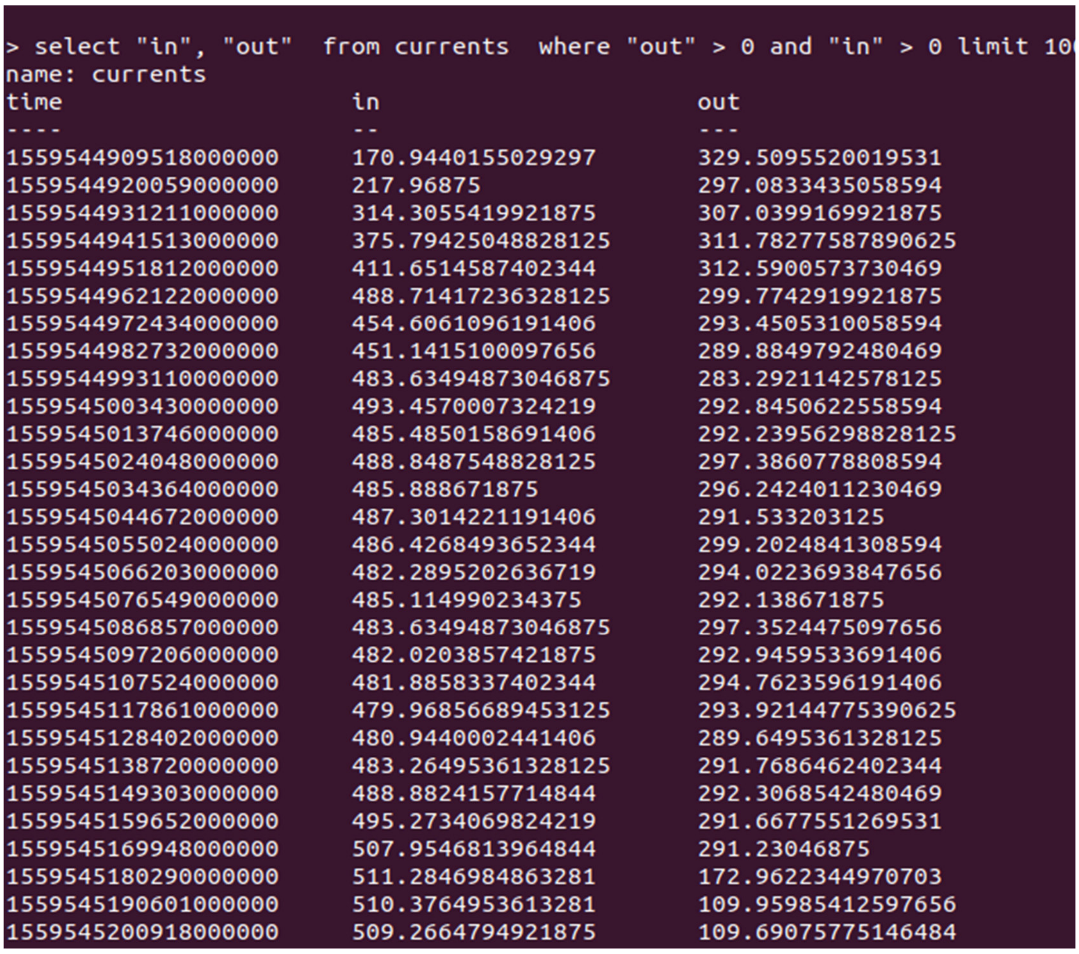
Figure 4: Data Alignment Mechanism

5.1. Data campaigns in FENIX

As data are produced form each step in the pilot plant network, these would be collected and aligned to the DSS Semantic Model and can thus be analysed in a uniform way. Output from this analysis will be given by the DSS higher-level modules (particularly the Analytics Services module); among the expected outcomes are a demonstration of cost reduction, patterns of usage that may be hint to optimum strategies and, of course, input to the DSS Engine.

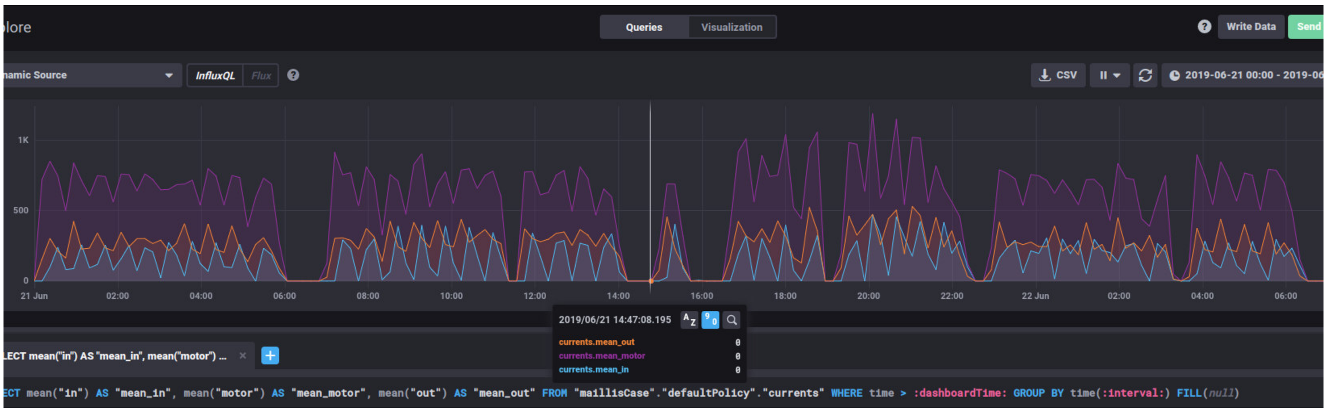
6. STORAGE AND QUEYRING OF DATA

After uplifting, data are forwarded to storage. This can be done either directly or via an event broker in case other modules (e.g. the DSS Analytics Services Engine) need to be notified. In either case, the data are readily accessible by exposing the relevant endpoints to the external modules; these endpoints can be used either to retrieve the data directly or to perform analytics.Figure 5 depicts one such example with the streaming data in raw format as these are queried by an InfluxDB client. Figure 6 depicts real time monitoring of streaming data in graphical format using a Grafana client. In this example, a set of historical datasets from the OROS dataset containing measurements of current in Ampere, were fed into the sensor adapter, thus simulating a stream. The data were smoothed using a mean function to avoid sudden jumps. The graphs depict three currents of three different motors; periods of downtime and periods of intense operation can be clearly discerned from the graphs. As sensors and historical data from FENIX pilots become available, these will be analysed in the same manner and be used for input by the DSS Analytics Services and the DSS Engine.



```
> select "in", "out"  from currents  where "out" > 0 and "in" > 0 limit 10
name: currents
time                   in                     out
----                   --                     ---
1559544909518000000    170.9440155029297      329.5095520019531
1559544920059000000    217.96875              297.0833435058594
1559544931211000000    314.3055419921875      307.0399169921875
1559544941513000000    375.79425048828125     311.78277587890625
1559544951812000000    411.6514587402344      312.5900573730469
1559544962122000000    488.71417236328125     299.7742919921875
1559544972434000000    454.6061096191406      293.4505310058594
1559544982732000000    451.1415100097656      289.8849792480469
1559544993110000000    483.63494873046875     283.2921142578125
1559545003430000000    493.4570007324219      292.8450622558594
1559545013746000000    485.4850158691406      292.23956298828125
1559545024048000000    488.8487548828125      297.3860778808594
1559545034364000000    485.888671875          296.2424011230469
1559545044672000000    487.3014221191406      291.533203125
1559545055024000000    486.4268493652344      299.2024841308594
1559545066203000000    482.2895202636719      294.0223693847656
1559545076549000000    485.114990234375       292.138671875
1559545086857000000    483.63494873046875     297.3524475097656
1559545097206000000    482.0203857421875      292.9459533691406
1559545107524000000    481.8858337402344      294.7623596191406
1559545117861000000    479.96856689453125     293.92144775390625
1559545128402000000    480.9440002441406      289.6495361328125
1559545138720000000    483.26495361328125     291.7686462402344
1559545149303000000    488.8824157714844      292.3068542480469
1559545159652000000    495.2734069824219      291.6677551269531
1559545169948000000    507.9546813964844      291.23046875
1559545180290000000    511.2846984863281      172.9622344970703
1559545190601000000    510.3764953613281      109.95985412597656
1559545200918000000    509.2664794921875      109.69075775146484
```

Figure 5: Raw Streaming Data stored in InfluxDB



```
ECT mean("in") AS "mean_in", mean("motor") AS "mean_motor", mean("out") AS "mean_out" FROM "maillisCase"."defaultPolicy"."currents" WHERE time > :dashboardTime: GROUP BY time(:interval:) FILL(null)
```

Figure 6: Retrieving Measurements of Current

7. CONCLUSION

This document presented all the modules developed for collecting legacy and sensor data, uplifting them and storing them in order to be used by the FENIX pilots and the higher-level modules of the DSS. As the pilot plants start exposing sensor data to the sensor fusion module, these will become readily available for consumption in order to generate analytics and recommendations which will then be used and evaluated by the pilots.

ANNEX A – EXAMPLE OF CODE IMPLEMENTING A ROUNDING BOLT IN STORM

```
public class RoundingBolt extends BaseBasicBolt {


    private static final long serialVersionUID = 1L;
    private static final Logger LOG = Logger.getLogger(RoundingBolt.class);


    @Override
    public void execute(Tuple input, BasicOutputCollector collector) {
        Double d = null;
        try {
            d = Double.parseDouble(input.getString(0));
            LOG.info("Retrieved measured value equal to: " + d);
            d = (double) Math.round(d * 10) / 10;
            LOG.info("Rounded value to: " + d);
        } catch (Exception e)  {
            LOG.info("Error rounding value: " + input.getString(0));
            return;
        }
        influxDB.close();
        LOG.info("Received message: " + input.getString(0) + " with length equal to: " +
input.getString(0).length());
    }
    @Override
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("message"));
    }
}
```

Figure 7: Rounding StormBolt

ANNEX B – EXAMPLE OF UPLIFTING USING MAPPING FILE

Figure 8 depicts a sample mapping entry for a sensor variable that corresponds to current. At the top level the entry denotes that it has to do with a “Measurement” entity, that hints to sensor data. The data type is “streaming” that denotes that these data will be collected real time. Following the data type the next list of attributes maps the InfluxDB entries to the Semantic Model.

- The first entry denotes that the name of the variable will be “current”
- The second entry denotes that the unit of this variable will be found from the tag “unit” of the InfluxDB entry. If the tag is unavailable or null the unit will be the default “fenix:ambere” unit, that is the Ampere unit that was defined in the DSS Semantic Model.
- The third entry, denotes that the “value” of this variable is in the default column of the InfluxDB entry (which is also named “value”)
- The final entry denotes, in similar fashion, that for marking the timestamp of the entry, the default InfluxDB “timestamp” name will be used; the difference is that this is now denoted explicitely.

```
<mapping>
  <package name="eu.projects.singular.fenix.dss ">
    <entity type="Measurement”>
      <data type="streaming">
        <fenix:name="current"> </fenix:name>
        <fenix:unit=”fenix:ambere”>unit </fenix:unit>
        < fenix:value></fenix:value>
        < fenix:timestamp>timestamp></ fenix:timestamp>
      </field>
    </entity>
        …
  </package>
</mapping>
```

Figure 8: Sample mapping entry