Computer Science and Information Systems 20(1):459–489 https://doi.org/10.2298/CSIS220301064B

Data-centric UML Profile for Agroecology Applications: Agricultural Autonomous Robots Monitoring Case Study

Sandro Bimonte[1], Hassan Badir[6], Pietro Battistoni[2], Houssam Bazza[6], Amina Belhassena[1], Christophe Cariou[1], Gerard Chalhoub[3], Juan Carlos Corrales[4], Adrian Couvent[1], Jean Laneurit[1], Rim Moussa[5], Julian Eduardo Plazas[4], Monica Sebillo[2], and Nicolas Tricot[1]

[1] Université Clermont Auvergne, TSCF, INRAE, France
{name.surname@inrae.fr
[2] University Salerno, Italy
{pbattistoni,msebillo}@unisa.it
[3] Université Clermont Auvergne, LIMOS-CNRS, France
gerard.chalhoub@uca.fr
[4] Universidad del Cauca, Colombia
{jcorral, jeplazas}@unicauca.edu.co
[5] University of Carthage, Tunisia
rim.moussa@enicarthage.rnu.tn
[6] IDS Team, Abdelmalek Essaadi University, Morocco
{houssam.bazza@etu.uae.ac.ma, badir.hassan@uae.ac.ma}

Abstract. The conceptual design of information systems is mandatory in several application domains. The advent of the Internet of Things (IoT) technologies pushes conceptual design tools and methodologies to consider the complexity of IoT data, architectures, and communication networks. In agroecology applications, the usage of IoT is quite promising, but it raises several methodological and technical issues. These issues are related to the complexity and heterogeneity of data (social, economic, environmental, and agricultural) needed by agroecology practices. Motivated by the lack of a conceptual model for IoT data, in this work, we present a UML profile taking into account different kinds of data (e.g., sensors, stream, or transactional) and non-functional Requirements. We show how the UML profile integrates with classical UML diagrams to support the design of complex systems. Moreover, We prove the feasibility of our conceptual framework through a theoretical quality assessment and its implementation in the agroecology case study concerning the monitoring of autonomous agricultural robots.

Keywords: Data Analytics, Internet of Things, Conceptual Modeling, UML Profile.

1. Introduction

In recent years, the Internet of Things (IoT) [29] has received much attention in multiple application domains, such as smart buildings and living, transport and mobility, healthcare, environment, energy, manufacturing, and *agriculture* and *agroecology* [4]. IoT represents a set of physical devices connected to the Internet that can generate, compute, store, and send data in real-time through different media (e.g., ZigBee, Wi-Fi, LoRaWAN) [1]. Moreover, the volume, heterogeneity and speed at which IoT can generate data classify them as a source of Big Data [29].

Technologies used for the implementation of IoT architectures have reached maturity. However, to the best of our knowledge, data-modeling methods for such architectures have not been well researched so far [29]. The **challenge in modeling IoT data** is caused by the fact that IoT data are: (1) *distributed and communicated over complex network architectures* (such as edge-fog-cloud) and (2) generated by a complex system. Such a system is typically composed of relational databases, NoSQL servers, and data stream management systems (DSMSs) besides the IoT. These components are implemented with various technologies supporting different programming languages and run on heterogeneous hardware (e.g., IoT devices, personal computers, and cloud servers). We refer to data generated in such a system as *polyglot data*.

IoT data are not only persistent but also transient. IoT data arrive into a system in the form of streams and are processed in real-time by streaming analytics applications [46] and Complex Event Processing (CEP) applications [53]. These applications monitor and discover trends and detect anomalies by means of continuous queries. Next, these data are typically stored into repositories such as *data warehouses* [52], *data lakes* [44] or *lakehouses* [54] to analyze them offline through OLAP (On-Line Analytical Processing) applications. Indeed, IoT real-time data are often combined with offline data to provide more advanced analysis [33].

Moreover, IoT applications are characterized by a geographically distributed deployment of devices and a network communication continuum over different layers (from the edge to the cloud) [38]. Therefore, Quality of Service features (QoS) plays a significant role in IoT data architectures, especially in the agricultural field of application, which is usually characterized by low quality communication networks. QoS can reflect some functional requirements, such as latency, which leads to a particular placement of data and computation over the different layers. For example, in the context of hard real-time applications, data and computation can be deployed at the edge level to improve performance.

Conceptual design of Information Systems (IS) has several advantages [43]. *First*, it allows to keep away implementation details and allows decision-makers and IS to exclusively focus application content and functionalities. *Second*, it provides a formal and non-ambiguous support used by decision-makers to validate their requirements. *Third*, it streamlines the implementation phase providing some technical guidelines (and sometimes also an automatic implementation). Although the conceptual design of data for IoT applications is crucial for their successful implementations, this topic has not been intensively researched yet [41].

Indeed, existing conceptual models do not allow to represent different data types issued from IoT in the same design framework. In addition, they do not support any QoS at the conceptual level. Therefore, the software engineering process for IoT-based applications is based on different conceptual models for each data type (stream, data warehouse, etc.), and QoS are taken into account at the implementation time. This implies that the merging phase of these different implementations is difficult or sometimes unfeasible.

This lack of design methodologies for IoT applications is evident in several domains, such as urban vehicles management (i.e., smart scheduling of traffic), health (i.e., real-time monitoring of physical and biological behavior of patients), logistic (i.e., smart affectation of human resources), tourism (i.e., enhance and optimize paths and stay) and also agroecology, which is the focus of this paper.

Nowadays, every organization, enterprise, and country must recognize the importance of new agricultural paradigms considering environmental, animal, and human health for sustainable development. In this line, agroecology is the main pattern to achieve this mandatory goal of humanity. The Food and Agriculture Organization of the United Nations defines agroecology as "*An integrated approach that simultaneously applies ecological and social concepts and principles to the design and management of food and agricultural systems. It seeks to optimize the interactions between plants, animals, humans, and the environment while taking into consideration the social aspects that need to be addressed for a sustainable and fair food system*" [7]. Agroecology evolves precision agriculture concepts, which mainly analyze crop-related data at detailed granularities, to consider more complex and global agronomic, social, economic, and environmental contexts [17]. Therefore, agroecological systems need a comprehensive approach, where versatile data types can be integrated and analyzed in multiple spatio-temporal dimensions.

Motivated by this lack of comprehensive solutions and by the formal support for data conceptual models provided by UML profiles, in [7] we proposed a UML profile for the data-centric design of agroecology IoT applications that considers some QoS network features, relevant at the conceptual level for end-users. Our UML profile, which is based on class diagrams, allows an *easy understanding and a formal representation* of these different types of data within a unique framework, which allows at the same time a coherent and global representation of all relationships between data. It is a crucial factor for the data-centric design of IoT applications. The different types of data have interactions among them in terms of data associations and network communications.

In this paper, we extend [7] in the following ways:

1. We present a design and implementation methodology centered on our UML profile. The main idea is to use the UML profile to define all data and non-functional requirements at a conceptual level in the same UML class diagram. Then, use these classes to define dynamic aspects of the application by means of other standard UML diagrams. In this way, our UML profile can be transparently adopted in all existing UML based software engineering development methodologies. To validate this feature, we show how classical UML-based software engineering design methodologies (such as [31]) can be used using our Class diagram UML profile. In particular, we leverage our case study with autonomous agricultural robots as an example. Besides, we detail how the UML Use Case and Activity diagrams can be used to derive and further detail the implementation of our Class diagram.
2. We provide a theoretical assessment of our UML profile quality, evaluating five quantitative metrics: *Reusability*, *Understandability*, *Well-structuredness*, *Functionality* and *Extendibility* according to [6,28].
3. We detail the technical implementation of each kind of data supported by our UML profile.

The paper is organized in the following way: Section 2 presents a real-world application in agroecology using IoT and autonomous robots. Section 3 presents our UML profile [7]. Section 4.1 presents the theoretical quality assessment. Section 5 shows the details of the implementation of the different supported data types (and underlying systems) using

[7] https://www.fao.org/3/i9037en/i9037en.pdf

our case study. Related work is shown in Section 6. Finally, Section 7 concludes the paper and proposes future work.

2. Motivation: Agricultural robots monitoring and scheduling case study

This section extends [7], presenting the motivation of our work by means of a case study, which will be also used in this paper to describe our proposal. In particular, the case study outlines the set of functional and non-functional requirements that must be supported.

The case study is based on the French I-SITE CAP2025 *Superob* project. The overall goal of the project is to develop and deploy an architecture for scheduling and monitoring field works of autonomous mobile robots used in agroecology practices. Autonomous agricultural robots represent an innovative solution for agroecology since they allow precise technical tasks and reduce environmental impacts.

With the advent of IoT, smart farming becomes a reality in the context of the agriculture domain. Farms are more frequently equipped with physical sensors [4] to acquire meteorological data such as rain, temperature, or soil moisture from the fields. Furthermore, autonomous robots are applied to handle technical operations, such as plowing [49].

As a business-like example of our *Superob* project, let us consider a scenario where a farmer needs to supervise the activities of some robots in a field. To this end, real-time data monitoring is necessary. Therefore, such a system must handle different types of data. In particular, we distinguish three basic data categories: stream, historical, and standard.

In this scenario, *Real-time streaming data* include, among others:

- *Trajectories of robots*, necessary to verify if a robot follows a scheduled trajectory, track the work in progress, and reschedule future tasks when necessary.
- *Meteorological data* (e.g., rain and wind data), necessary to check whether a given robot task can be done, e.g., some tasks such as spraying cannot be run when the wind is too strong. These data can be provided by some external weather services or by meteorological stations installed in the field. The choice depends on the needs and economic possibilities of the farmer. Using sensors provides more precise spatial scale data, but it is more expensive than free or commercial external meteorological services. In any case, these two different information sources must provide a minimal subset of equal attributes, such as air temperature and humidity.
- *Odometry robot data* (i.e., mechanical robot data), necessary to determine whether robots are experiencing any mechanical problems.
- *Scheduling data* (i.e., demands from farmers), necessary to define the organization of robots' tasks.

Historical data are crucial for decision-making. Analytical queries analyze such data. For example, historical data corresponding to the same robot in the same field and its technical operations allow comparing the current work to the past ones, to decide if the robot has abnormal behavior.

Finally, *standard data* are needed to complement real-time and historical data. Examples of standard data include: lists of plots (with their geometries and basic data), as well

as robots characteristics. These data represent the contextual information associated with other data and play the role of dictionary data.

Decision-support applications usually provide *computations* over data to calculate new indicators. In our scenario, we compute continuous queries on real-time data to create meteorological, robots-fault, and delay alerts. These alerts must also be stored since, as described above, they are useful for the decision-making process.

Data type requirement: The aforementioned scenario allows us to state that *agroecology IoT applications must cope with: (1) complex spatio-temporal data (such as robots trajectory), (2) stream data (such as weather information and robots data generated in real-time data), and (3) historical data.*

Non-functional requirements refer to systems and constraints, such as time computation or quality. In our context, since decision-makers must supervise their agricultural practices in real-time, they must agree to a set of non-functional requirements that must be supported by the system. In some cases, such as the computation time for robot faults, they must be involved in the definition step of these constraints. Therefore, we consider that taking into account non-functional requirements about data at the design step is a mandatory issue for agroecology IoT-based applications.

Moreover, the discussed data are deployed over the network architecture presented in Figure 1, which is typical for rural agricultural areas. As most farms are located in rural areas, the cellular network coverage might not be enough to ensure the QoS, which can be considered as network non-functional requirements, required by the application. Also, the use of cellular networks induces an additional cost for every node (sensors and robots) associated with the network. Consequently, they communicate and send their data through a standard local Wi-Fi connection in this architecture. They could also use other wireless technologies like *Zigbee* or *LoRaWAN*. Data from robots and sensors are thus sent to a workstation deployed on the farm. The workstation has a standard internet connection, through which data are sent to the cloud containing complex decision-making applications.

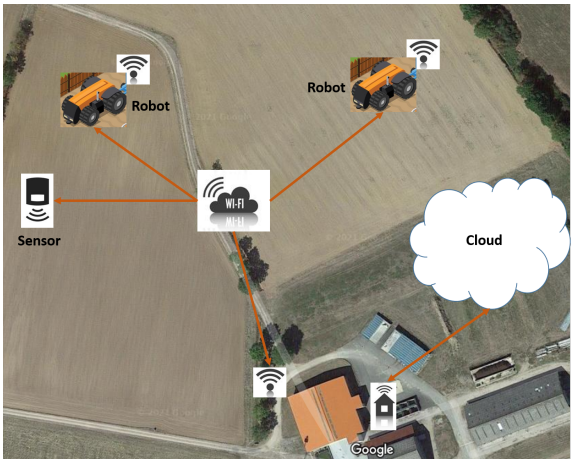

Fig. 1. Network communication example in a field

The network must ensure a particular QoS to meet the requirements of decision-makers. For example, Internet communication (e.g., an ADSL network) available in the farm must be fast enough to provide real-time data exchange with a cloud (e.g., online adjustment of scheduled tasks). The control of faulty robots using odometry data requires very low latency for real-time communications between a farmer and a robot. This latency cannot be satisfied on the end-to-end link with cloud servers, and thus it should be locally established. Hence, the local wireless network (Wi-Fi network) should be designed to answer these performance needs.

Non-functional requirements requirement: For the previously mentioned reason, *we argue that network performance indicators as well as data non-functional requirements must be integrated into the design of data-centric agroecology IoT applications*.

Depending on the network capacities of an Internet connection and a local Wi-Fi network, the distribution, storage, and processing of data would be modeled differently. For example, tasks requiring a high data rate and low latency (like remote control of faulty robots) should be executed on the farm and not on the cloud in case that the Internet connection does not support such a QoS.

Finally, let us note that data produced and consumed by the architecture components are strongly related to each other, which must be reflected in a data model. For example, a robot during its work must be associated with the plot where it is working, and it needs access to meteorological sensors' data of the plot.

Network communication also plays a crucial role in disseminating the information obtained through data analysis once they have been analyzed, either on a farm workstation or in the cloud. The information must reach its consumers, i.e., decision-makers of various roles who may be geographically distributed. To this end, visualization tools must be able to present the information asynchronously from multiple sources, producing data at different rates. Furthermore, some of the information must be communicated in real-time, e.g., rescheduling a robot that lost its trajectory or is malfunctioning.

From the above described example, we can conclude that the different actors involved in the design and development of agroecology applications are:

- *Agroecology stakeholders*, who define the application's requirements.
- *Information systems experts*, who are in charge of the implementation of the different systems to store and provide standard, historical and stream data.
- *IoT experts*, who provide the implementation of different IoT devices (e.g., sensors, robots, etc.).
- *Network experts*, who set up and configure the communication networks.

High quality model requirement: Since different actors are involved in the design step, the formalism used must be effective and grant important quality issues, such as understandability and reusability.

The usage of classical design and implementation methodologies is depicted in Figure 2. For each kind of data, a conceptual design step, and then its implementation, are **separately** applied. Then, the communication network configuration is provided. Finally, these different systems are coupled together to finalize the application deployment. Usually, this *merge* step raises several problems due to:

- *At design step*: Requirements are not well and exhaustively defined. Indeed, agroecology stakeholders must exchange separately with other actors. Therefore, it does not prevent them from having a global and unique vision of the defined requirements.

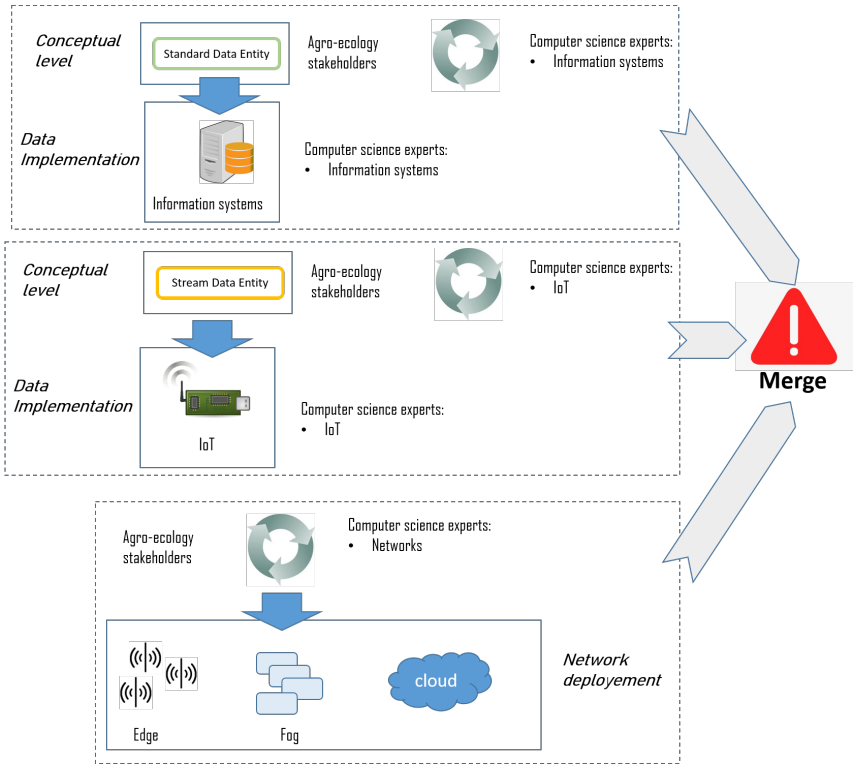

Fig. 2. Existing methodologies

– *At implementation step*: the risks related to (i) Possible incompatibility of the different systems implementation and the communication network communication constraints, and (ii) Manually generation of code representing associations among the different data, which is usually the source of technical and conceptual translation errors.

Integrated design and implementation requirement: The design step must be supported by a unique formal framework supporting different kinds of data and non-functional requirements and that makes transparent all implementation issues related to the usage of different technologies. Moreover, this formal framework could be used with other existing design methodologies to represent the dynamic aspects of the system.

Therefore, *there is the need for a unique data-centric conceptual model that allows all involved actors to exchange information about the requirements of the agroecology applications supporting different kinds of data and network communication issues*.

An example of the implemented web interface application is shown in Figure 3. The details of the implementation are shown in Section 5. Figure 3 clearly shows the different kinds of data involved in the application:

– meteorological data, which represent air humidity and temperature. These data are issued from the meteorological station.
– odometry data, which represent the robot speed. It is represented with a line chart.

- real time trajectory data. These data represent the real time position of the robot with a red line. The predefined trajectory is represented with a yellow line. A bullet point can be used to also visualize other odometry data in real time.
- background data. The visualized map is issued from google map or any other map server that could be used.
- video data collected by drone. This video is issued from a drone used in the experiment we have done.

3. UML Profile

In this section, which extends [7] with two new subsections,we present our UML profile for data-centric agroecology IoT applications. In Section 3.1, we present an overview of our UML profile, and then we detail the data and associations' representations.

3.1. Overview

Our UML profile provides a graphical and formal notation for functional requirements (in terms of data). A UML profile provides a generic extension mechanism for customizing UML models for particular domains and platforms. It is defined using stereotypes, tag definitions, and constraints applied to specific model elements, like Classes, Attributes, or Operations. We opt for an extension of UML elements of class diagrams since they are the de-facto standard to represent data.

Our UML profile allows designing all different kinds of data, and their associations, with the same UML Class diagram. Moreover, some network communication features can also be added inside this Class diagram. In this way, the design step of the agroecology application involves all the involved actors (agroecology stakeholders, information systems, IoT and network experts) at the same time. They share the same graphical formalism to exchange among them, which allows to avoid the *merge* step problems described in the Section 2. Moreover, the usage of Class diagrams allows using our UML profile with other tools provided by UML for the definition of functional and non-functional requirements, such as Use Case, Activity and Sequence diagrams (as shown in Section 4.2).

The design and implementation methodology for IoT applications based on our UML profile is depicted in Figure 4. The first step consists in the design of a conceptual model for all data involved in the applications and the associated QoS. This step concerns all the actors involved in the system (i.e. decision-makers - agroecology experts in our scenario, IoT, information system and network experts). This step can be iterative, and can include other UML diagrams to represent dynamic aspects of the application. Once an agreement about the conceptual model is found, the real implementation can be provided in different systems (sensors, database systems, etc.) for each data (by IoT and information system experts). Finally, the network communication is configured by network experts. In our approach, moving from the conceptual model to the implementation steps is feasible and do not require the intervention of decision-makers, since all the involved actors have reached an agreement about all data, IoT devices, and network configurations that will be used by the applications using the UML diagrams. This avoids the merge problems described in the previous section.

Fig. 3. Application user interface

Figure 5 shows the meta-model of our UML profile. In the next of this Section, we detail each element of the meta-model[8].

[8] A video describing the usage of the UML profile with Eclipse can be found here `www.youtube.com/watch?v=uTRewVj_eDs`

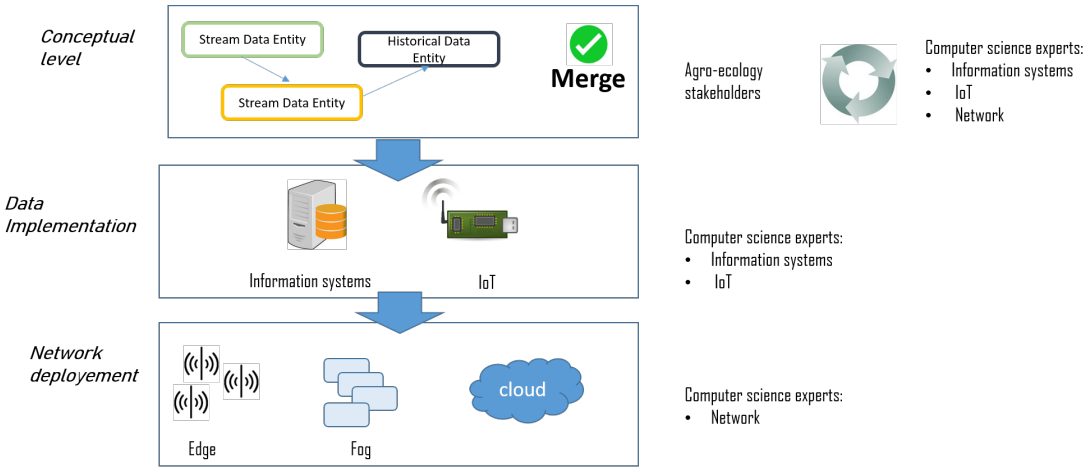

Fig. 4. Our approach

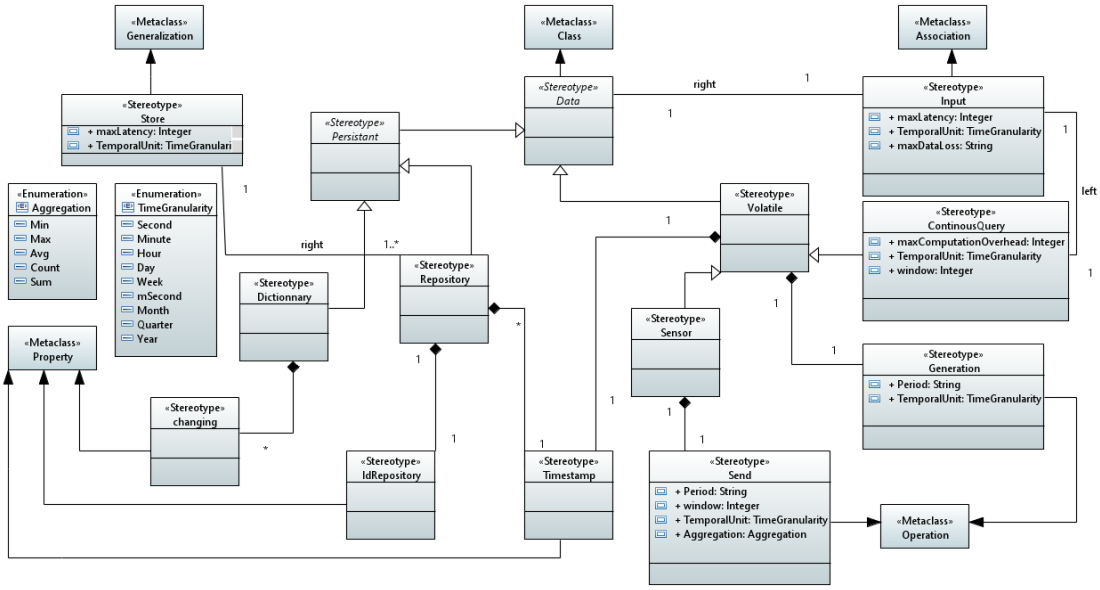

Fig. 5. The meta-model of our UML profile

3.2. Data

Data are classified into two main groups, namely, `Persistent` and `Volatile`, which are Class stereotypes. The `Persistent` stereotype is specialized in `Dictionary` and `Repository`.

`Dictionary` represents transactional data (i.e., standard data) that can be deleted, updated, and inserted in an On-Line Transaction Processing (OLTP) system. `Dictionary`

can include some attributes stereotyped as `Changing`. This stereotype means that attribute values can be updated, contrary to other attributes whose values do not change. The following associated Object Constraint Language (OCL) rule states that the attribute must not be changed (`isReadOnly=true`). Moreover, the `Dictionary` class must provide some attributes that uniquely identify its instances. This constraint is represented using the following OCL on such attributes: `isUnique=true`.

An example is shown in Figure 6, where the `Dictionary` stereotype is applied to *Robot*. This class presents (1) some standard attributes (e.g., *Name*, *SpeedMax*, *Weight*), and (2) some `Changing` attributes, like *Available*, which indicates when a robot is available for a particular task or is booked for another task within a given time slot.

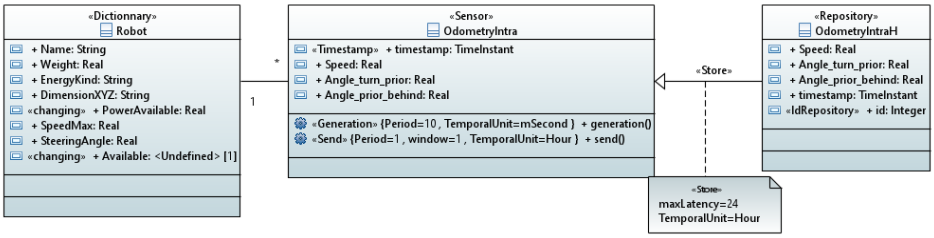

Fig. 6. Examples of *Dictionary*, *Sensor*, *Repository*, and *Store* association examples

`Repository` represents read-only historical data with the following characteristics:

- Attributes of the `Repository` class cannot be updated; only new values can be inserted. This constraint has been defined with OCL in the following way: `self.ownedAttribute->select(m|m.isReadOnly=false)->size()=0`.
- An instance of the `Repository` class cannot be deleted; it can only be inserted.

Moreover, `Repository` includes one attribute with stereotype `IdRepository` that uniquely identifies a datum in the collection of historical data (OCL: `ownedAttribute->select(m|m.oclIsTypeOf(IdRepository))->size()=1`). Finally, to model the temporality of the historical data represented by `Repository`, a `Timestamp` stereotype attribute is added, with an OCL constraint that forces it to have the `TimeInstant` type (OCL: `type.name='TimeInstant'`). Thus, `Repository` data represents historical data used for analytical purposes, such as OLAP or Machine Learning applications. An example is shown in Figure 6, where *OdometryIntraH* represents odometry historical data of robots.

`Volatile` represents data producers. These data are not permanently stored, and are characterized by a frequency generation represented by an operation with the `Generation` stereotype. `Generation` has two tagged values:

- `Period` that represents a temporal generation frequency, e.g., every second. In case of data generated on-demand, `Period` also accepts the *onDemand* value.
- `TemporalUnit` is the temporal granularity of `Period`. It takes values from enumeration `TimeGranularitiy`, e.g., second, minute, hour. This enumeration can be easily extended with other temporal types.

Moreover, `Volatile` also has one `Timestamp` attribute representing the time of the data generation.

The `Volatile` class is specialized into another class, called `Sensor`, which represents volatile data that are generated by physical sensors. `Sensor` extends `Volatile` with the `Send` operation, which represents the logic used for sending the data. It has the same tagged values of `Generation` (`Period`, `TemporalUnit`), and the following additional ones:

- `Window` represents a temporal window used to collect and process data before being aggregated and sent.
- `Aggregation` represents the aggregation function used on the collected data in the window before being sent. It takes values from enumeration `Aggregations`, e.g., Sum, Avg, Count (other aggregation functions can extend this enumeration).

It is crucial to specify these particular data sources at the design time since sensors must send data through a communication network, which can have substantial impacts on the system implementation.

An example of `Volatile` data are represented by the instances of class *Demand* -illustrated in Figure 7-B. This class model the activity requests of working tasks performed by a farmer. The instances of this class are generated on-demand. Consequently, tagged value `Period=onDemand`.

An example of sensor data is shown in Figure 7-A. It represents meteo data acquired by a sensor. Data (*wind, rain, temperature*, etc.) are collected each minute (`Period=1` and `TemporalUnit=minute`). Then, averages in a moving 10-minutes window are calculated.

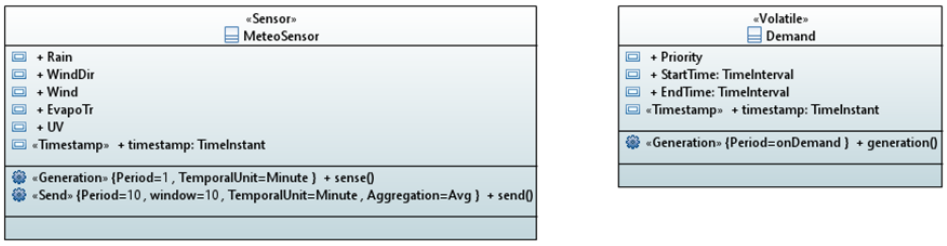

Fig. 7. Example instances of *Sensor* (**a**), and *Volatile* (**b**)

Commonly, a continuous query is executed over a data stream.

A continuous query is a query, which is re-computed continuously. For example, the query "Each minute, give me the average temperature of the last 10 minutes" will return different results depending on the current time.

In our UML profile, the stereotype `ContinuousQuery` represents a continuous query. It extends the `Volatile` stereotype with:

- `ComputationOverhead` tagged value, which represents the maximum time to compute the query.

– `Input` directed association, which represents the input data used for the query. `Input` has two tagged values: `maxLatency` and `maxDataLoss`. `maxLatency` represents the maximum tolerated time for input data to be transmitted into the system that implements `ContinuousQuery`. `maxDataLoss` represents the percentage of data that can be lost. These QoS constraints are issued from the application logic and come from the fact that data are generated in different points of a network, as described in the IoT architecture. Other network performance constraints exist; yet, they correspond to non-functional requirements (NFR), but not to the application logic. For instance, bandwidth is associated with a particular implementation of attribute data types (in terms of bytes used). Such NFR constraints should be represented at the Platform-Specific Model level following the Model-Driven Architecture, while our UML profile would correspond to the Platform Independent Model level.

The NFR are used to guide the implementation of the system. They impact the choice of the components of the system. For example, a low `ComputationOverhead` for the DSMS component implementing the query could necessitate a distributed DSMS, or a low `maxLatency` could lead to the use of a new communication network such as 5G instead of ADSL. If the NFR are not met temporarily then the multi-representation solution can be applicable. Multi-representation has been defined for classical data, and in particular for Geographic Information Systems [56], as different representations and computations of the same entity data according to different rules.

Figure 8 shows an example of `ContinuousQuery`. *AlertDelayQuery* computes in real-time the delay of a robot according to its predefined trajectory. It takes as inputs: *Point-Time*, which represents the real time position of the robot, and *TrajectoryRef*, which represents the planned trajectory. The tagged value of the *Input* association states that these GPS data must be received in real-time for the alert delay computation. Moreover, *AlertDelayQuery* is computed each minute using the last 5 minutes of received data, and 5% of GPS data can be lost, contrary to *TrajectoryRef* that cannot be affected by data loss (i.e. all data of the trajectory of reference must be present). End-users define the configuration of *AlertDelayQuery* parameters.

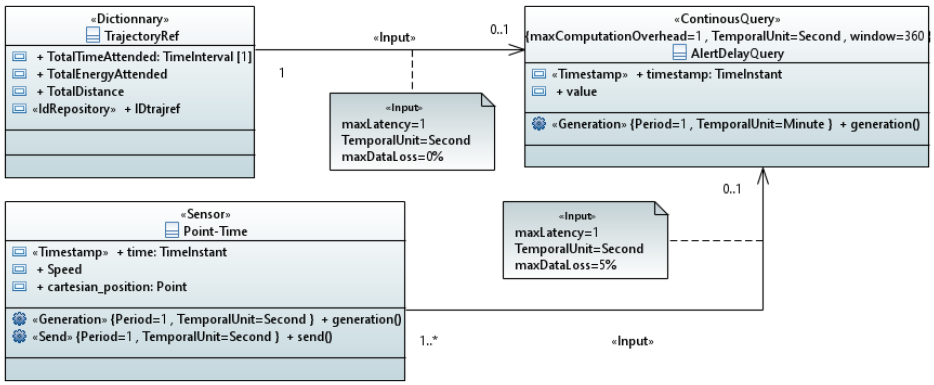

Fig. 8. An example of stereotype *ContinuousQuery*

Other kinds of queries are compatible with our approach. For example, it is possible to provide multidimensional queries (i.e. read only on-line queries over warehoused data - such as "*What is the average temperature per plot and month*?"), or transactional queries (i.e. update and write queries - such as "*Update the location of the sensors A*") [9].

3.3. Associations

This section describes how `Volatile` and `Persistent` data can be transparently associated to define a single coherent model.

From Figure 8, we can notice that any data can be associated with `ContinousQuery` via `Input` association. Moreover, `Volatile`, `Sensors`, and `ContinousQuery` can be associated with `Repository` data via the `Store` association. This association means that initially volatile data are made persistent by using the `Repository` class. As for the `Input` association, it has a `maxLatency` tagged value. This value represents a maximum time within which data must be stored in a repository. Since volatile data could be sent through a communication network they cannot be stored immediately, thus we use `maxLatency` value.

For example, Figure 6 shows that data collected by *OdometryIntra* into the robots (odometry data collected 100 times per second, and sent every hour) are stored into the `Repository` class *OdometryIntraH*. The `Store maxLatency` value is 24 hours since these data are stored in a data warehouse refreshed every 24h.

The association between `Store` and `Repository` is a generalization, because the `Repository` class must include in its structure all the attributes and associations of classes `Volatile`, `Sensor`, and `ContinuousQuery`. Moreover, `Repository` must not present methods of `Volatile` (OCL: `ownedOperation->size()=0`). Therefore, the `Store` association represents a total cloning operation of the `Volatile`, `Sensor`, and `ContinuousQuery` data in persistent storage.

Let us consider the example of Figure 6 again. If a persistent storage stored only the values of the odometry attributes, such data would be incomplete. Note that *OdometryIntra* is associated with `Robot`. Without the associated robot that generated these data, it would not be possible to identify the robot that has generated such odometry data.

To conclude, this data-centric representation of all kinds of data and queries allows us to associate all these data among them without considering if the data is classical data, or sensor data or stream data or data resulting from computations.

Therefore, our proposal satisfies the *Data types* and *non-functional requirements* described in Section 2.

4. Assessing proposed UML profile

In the above section, we have pointed out how our UML profile can be easily used to represent different kinds of data and non-functional requirements, as described in section 2. Therefore, in this section, we provide some theoretical and practical evaluation to show how the other defined requirements are supported by our proposal.

4.1. Assessing the quality of the proposed meta-model

In this section, we provide a quantitative validation of the quality of our UML profile following the framework proposed in [28], in order to show how our proposal satisfies the *High quality model requirement* identified in Section 2. The framework proposed in [28] allows measuring the quality of a metamodel using five metrics calculated from the metamodel with a three step process. First, the following metrics are computed:

- ANDM: the average number of direct associations between metaclasses.
- ANM: the average number of attributes.
- ANMC: the average number of direct association between a metaclass and other kinds (operation, property,...).
- ANR: average number of OCL constraints.
- NOH: the number of inheritance hierarchies
- ADI: the average depth of inheritance trees
- ANA: the average number of direct inheritance between metaclasses
- NAM: the number of abstract metaclasses
- NCM: the number of concrete metaclasses

Then, using the above presented metrics, some global metrics (such as modeling concepts size, abstract metaclass size, intension, coupling, ...) are computed. For instance, coupling that means the level of interdependence between the classes of a diagram, is computed as the sum of ANDM and ANA.

Finally, five ultimate metrics (calculated using the metrics of the second step provided.

- *Reusability*: it measures the ability of a metamodel's components to contribute to the definition of different metamodels (e.g., in other application domains).
- *Understandability*: it represents the degree of ease to understand and to use the content of a metamodel by end-users.
- *Functionality*: it measures the number of concrete metaclasses which reflect the strength of modeling ability of a metamodel.
- *Well-structuredness*: it represents how a meta-model is well-structured by measuring the structure quality of its architecture by means of its metaclasses.
- *Extendibility*: it measures the ease to add new modeling element to a metamodel. It is computed as **0.2 × Coupling + 0.3 × (Modeling concepts size + Abstract metaclass size))):**

Moreover, these quantitative measures enable quality comparisons against other metamodels. Indeed, [6] evaluates and compares more than 2500 UML metamodels from the literature using this framework ([28]). Similarly, we compare our UML profile with those works using the percentile rank. In this way, we can assess if our metamodel is outstanding, regular, or particularly bad on each quality measure.

Figure 9 shows the results of such comparisons. When compared to more than 2500 UML metamodels analyzed in [6], it is evident that:

- Our profile excels in Understandability and Well-structuredness, which are really important due to the diversity of actors involved in the design phase as described in Section 2.

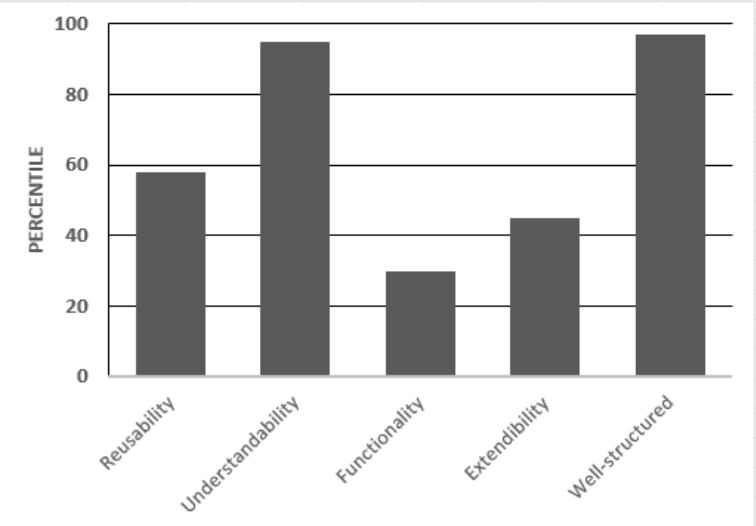

Fig. 9. Comparison our metamodel according to the framework of [28], in the context of all the meta-models considered in [6]

- Its Reusability and Extendibility are around the median. Therefore, it will not demand significant efforts to add new data elements, such as multimedia data.
- Only the functionality is below average, but it is not particularly bad. A low functionality usually means that the range of applications that our metamodel covers is narrow. However, we must consider that we use highly abstracted concepts in our profile, and thus each profile element relates to multiple implementation possibilities.

Considering the outcomes from this theoretical assessment, we infer that the quality of our UML profile is appropriate for the design of agroecology applications.

4.2. Assessing the integrated design

In the next, we describe how our UML profile can be used in a classical software UML based design methodology. For simplicity we exclusively focus on the class diagram represented in Figure 6. UML use case and activity diagrams are well recognized as effective tools for collecting and formalizing requirements [32]. These diagrams are then used to deduce Class diagrams. In our case study, one main functionality of the monitoring system is the analysis of historical odometry data, which are collected by robots in the field. The use case diagram showing this task is represented in Figure 10. It presents two actors, the robot that collects the odometry data in real time, and the repository system (i.e., database system) that stores all data collected by all robots during their work (i.e., historical odometry data). Therefore, the following classes can be deduced: *Robot*, *OdometryIntra* (for real time odometry data) and *OdometryIntraH* (i.e., for the Repository actor).

According to this use case, the associated activity diagram is shown in Figure 11. From this activity diagram, it is possible to deduce the need for (i) a generation method for *OdometryIntra*, (ii) a method that represents sending data over the network (from robot

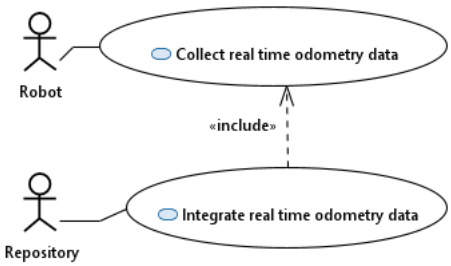

Fig. 10. Use case diagram

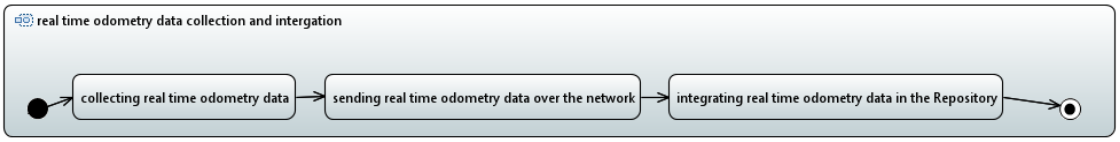

Fig. 11. Activity diagram

to the repository) (*send()*), and (iii) a store method to represent the integration of real time odometry data in the repository (i.e. *OdometryIntraH*).

Sequence diagrams can be used to express time constraints. Therefore, the need for the decision-makers to be able to analyze the last 24 hours collected odometry data could be expressed by means of a sequence diagram. In this work, in order to keep the UML models as simple as possible, instead of using a sequence diagram, we opt for using our UML profile since it is possible to represent this constraint using the `Store` association and its tagged value. Therefore, `Store` association represents the integration of real time data into historical data and the temporal constraint. At this point, using the use case and the activity diagrams we have easily obtained a skeleton of the three main classes and their associations of Figure 6. Finally, discussing with the agroecology stakeholders, the IoT, information systems and network experts can complete the class diagram to obtain the final one depicted in Figure 6. Indeed, using this skeleton of class diagram makes it more simple for agroecology stakeholders to define the details of each class, and therefore the choice of the right stereotype. At the end, by means of our UML profile, agroecology stakeholders are aware about the volatile (or not) character of data involved in the system and the fact that data are exchanged over a communication network.

From the above described example, we can conclude that our proposal supports the **Integrated design and implementation requirement** described in Section 2.

5. Implementation

This section presents the implementation in a commercial CASE tool, and how each type of data of our agricultural case study is implemented (Section 5.2), and we detail its corresponding IoT architecture (Section 5.3).

5.1. CASE tool implementation

In this section, we present the implementation of our UML profile by using Papyrus,an open-source software for UML modeling based on Eclipse. Papyrus supports the creation of UML profiles by specifying instances of the different UML meta-elements (e.g., stereotypes of properties, classes, operations, or OCL constraints). Papyrus allows checking OCL constraints at design time. For example, let us consider Figure 12. It shows how Papyrus checks that the constraint: (OCL:`ownedAttribute->select(m|m.oclIsTypeOf (IdRepository))->size()=1`)) for `Repository` (which indicates `Repository` must include one attribute with stereotype `IdRepository`).

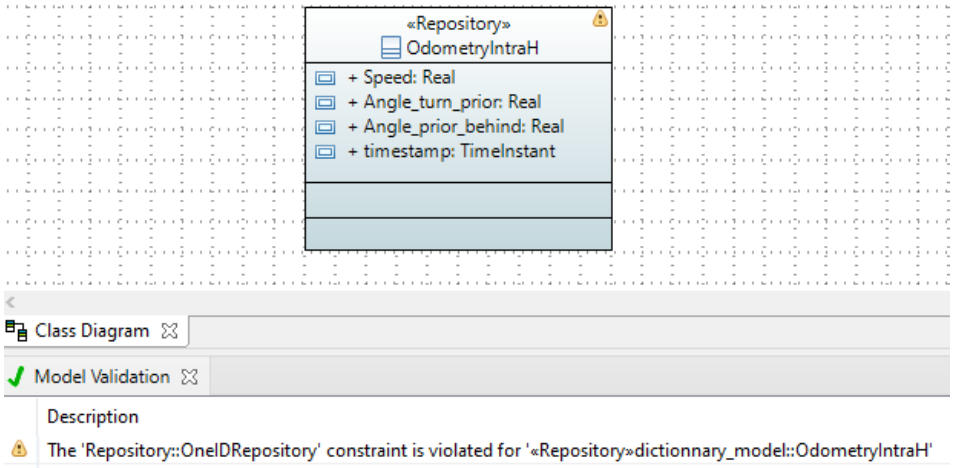

Fig. 12. Example of OCL constraint check with Papyrus

The UML profile implementation is available as open source project [9].

5.2. Data implementation

The implementation of our case study requires a complex digital ecosystem. Despite a uniform representation of data at the conceptual level, the different kinds of data must be generated and handled by diverse subsystems. For instance, `Sensor` and `ContinousQuery` data require an implementation using programming languages and Data Stream Management System. `Persistent` data either in a classical storage system (such as a relational database) or in a novel storage system (such as NoSQL systems when scalability is needed) can be deployed.

Consequently, in our case study, each component of the digital ecosystem have a particular implementation.

[9] https://www6.inrae.fr/tools4bi/Design/A-UML-Profile-for-Agroecology-data-centric-applications-design
We will provide it for download after the acceptance of the paper

Meteorological data (represented by the `Sensor` *MeteoSensor* in Figure 7-A) is implemented in an IoT device running *RIOT OS* [10]. RIOT is an open source operating system that supports several low-power IoT devices, micro-controller architectures (32-bit, 16-bit, 8-bit), and external devices. Applications for this OS are written in C and must specify the behaviour of each involved device [5]. Figure 13 shows two fragments of the code associated to *MeteoSensor* (Figure 7-A).

```
1.    /* Code for thread 'AirTemperature_avgThread' */
2.    /* Stack of memory for thread 'AirTemperature_avgThread' */
3.    static char AirTemperature_avgThread_memstack[THREAD_STACKSIZE_MAIN];
4.    /* Thread 'AirTemperature_avgThread' */
5.    /* Thread to sense and transform one variable */
6.    static void *AirTemperature_avgThread(void *arg)
7.    {
8.          (void)arg;
9.          while (1) {
10.                 AirTemperature_avg_dataStruct internalData;
11.                 //gather the variables
12.                 for (int16_t w=0; w< 10; w++) {
13.                           lpsxxx_read_temp(&lpsxxx, &internalData.avgTemp_src[w]);
14.                           internalData.avgTemp_src_lastplace = w;
15.                           xtimer_sleep(60);
16.                 }
17.                 //Apply average with order 0
18.                 // Aggregation: 'average'
19.                 int16_t sum = 0;
20.                 for (int8_t  i=0; i<=internalData.avgTemp_src_lastplace; i++){
21.                           sum += internalData.avgTemp_src[i];
22.                 }
23.                 int16_t average = sum / (internalData.avgTemp_src_lastplace + 1);
24.                 internalData.avgTemp_src[0] = average;
25.                 internalData.avgTemp_src_lastplace = 0;
26.                 // Save the internal data into the public struct:
27.                 mutex_lock(&public_AirTemperature_avg.lock);
28.                 public_AirTemperature_avg.data = internalData;
29.                 mutex_unlock(&public_AirTemperature_avg.lock);
30.         }
31.         return 0;
```

Fig. 13. Sensor implementation

The first code fragment (Figure 13) is the sensing and aggregation thread. To begin, this thread samples the plot temperature 10 times with a periodicity of one minute (i.e., during 10 minutes). Then, it calculates the average temperature of the plot of the last 10 minutes and saves it as a public variable. Finally, the thread process starts again to run indefinitely.

It is important to note that the data implementation strictly follows its conceptual definition. The IoT code (Figure 13) senses data every minute, and calculates the average and sends the data every 10 minutes as specified in *MeteoSensor* (Figure 7-A).

Odometry data (*OdometryIntra* in Figure 6) are implemented in Python in the *Fleet of Robots* using Robot Operating System (ROS). Besides, robots have tasks, trajectories, and timing constraints (e.g., indicated speed).

[10] https://www.riot-os.org/

`Persistent` data (e.g., *OdometryIntaH* in Figure 6) are stored in the relational spatial DBMS PostGIS. These data are further loaded into a *Data Warehouse* implemented in Mondrian and JRubik to analyze them.

The Data Warehouse storage is provided by PostGIS. It is important to note that the implementation of `Persistent` data needs some particular SQL statements. Indeed, attributes with the `Changing` stereotype are classical ones, contrary to the other ones that cannot be updated. This constraint is implemented in SQL with a trigger on the UPDATE SQL statement. An example for the *name* attribute of the *Robot* class is shown in Figure 14.

```
1. CREATE OR REPLACE FUNCTION not_changing()
RETURNS trigger AS
$BODY$
    BEGIN
        RAISE EXCEPTION 'no way';
    END;
$BODY$
  LANGUAGE plpgsql VOLATILE

2. BEFORE UPDATE OF Name
  ON Robot
  FOR EACH ROW
  EXECUTE PROCEDURE not_changing();
```

Fig. 14. SQL implementation example

The *AlertDelayQuery* continuous query (Figure 15) is implemented in Scala (the Sedona framework, which is a spatial extension of Apache Flink). This query joins GPS data coming from the robots (`Sensor` *Point-Time*) with data stored in PostGIS (`Dictionary` *TrajectoryREF*) (Line 1).

Then, the delay is computed (Line 2). Data is collected in a window of 1 minute (Line 3), and each 1 minute data is sent using the average aggregation function (Line 4).

```
1. val results=jdbcDF.as("a").join (TrajDf.as("c")).where("a.idDB
   = c.id") //Retrieve the table with the reference trajectory using the full join query

2. val delay=
   results.withColumn("DiffInSeconds",col("stampDB").cast(LongType
   ) - col("stamp").cast(LongType)) // compute the delay and add the result to
   the new column.

3. val windows1 = delay .groupBy(window($"stamp", "60 second", "60
   second"), $"id",$"lat",$"long", $"idDB",$"latDB",$"longDB",
   $"DiffInSeconds") //Group the trajectory data by window and delay.

4. val aggregatedDF1 = windows1.agg(avg("DiffInSeconds")) //aggregate
   the window results and calculate the average delay of each window group.
```

Fig. 15. Spark Streaming implementation example

Moreover, associations between `Dictionary` and `Repository` can be simply implemented using foreign keys in a relational DBMS or integrity constraints in a NoSQL DBMS. However, such mechanisms do not exist for `Volatile` data (`Sensor` and `ContinousQuery`). They require an ad-hoc method: the data structure sent by sensors must include their identifiers, such as previously described above for the meteorological sensor.

5.3. Multi-layer network architecture

This subsection is issued from [7]. Inspired by the Lambda reference architecture for IoT applications [48], we propose an architecture to host the technologies that handle the data required in our case study. It is composed of three main layers: *Field*, *Farm*, and *Cloud* (Figure 16). The *Field* and *Farm* layers are implemented in each farm. In contrast, the *Cloud* layer is implemented only once for any number of farms.

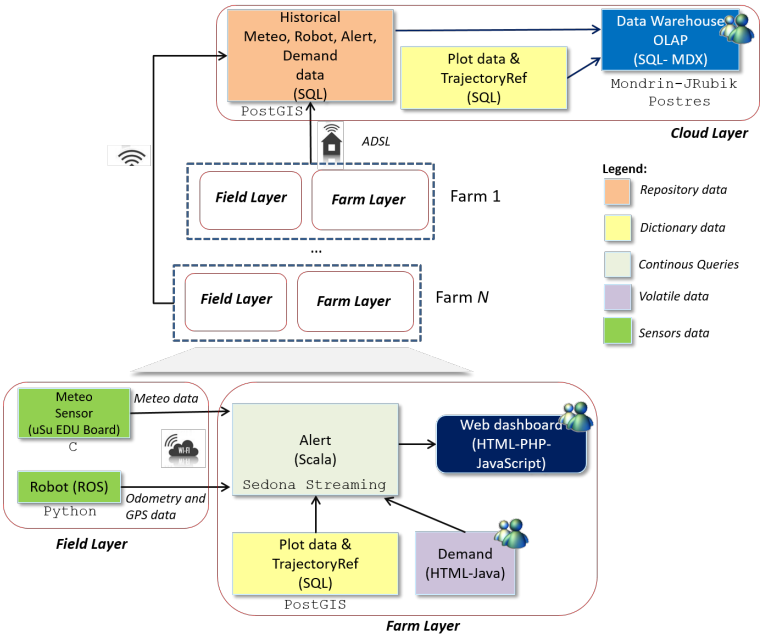

Fig. 16. Architecture implementation - data mapping

The *Field* layer represents different data sources deployed in the field (i.e., `Sensor` data). In our case study, it is composed of *MeteoSensor*s and *Robot*s, which provide data and execute specific tasks. These IoT devices do not have direct Internet access on the fields (there is no cellular network coverage). Thus, we deploy a Wi-Fi network to collect these data at the *Farm* layer, which connects to the Internet through ADSL.

However, ADSL may not guarantee the QoS required by the `ContinuousQuery` of our case study. For instance, the `Input` association between *AlertDelayQuery* and

TrajectoryREF (Figure 8) requires a maximum latency of 1 second with no data lost. Therefore, we implement these queries in the *Farm* layer. Moreover, we host a DBMS for *TrajectoryREF* in the same layer (*Farm*) since this allows for a local connection to the DBMS with less latency and data loss issues. Finally, the *Farm* layer only sends processed data streams to the *Cloud* layer.

The *Cloud* layer implements the storage of `Repository` and `Dictionary` data coming from all the farms. This layer hosts a data warehouse that analyzes the historical odometry data of the robots, providing an inter-farm analytical vision of all the available data.

This way, our case study shows that the proposed UML profile can effectively represent the multiple kinds of data present in a complex IoT-based application. The data was implemented in different languages, systems, and infrastructures that must successfully communicate and cooperate to provide useful decision support.

6. Related Work

In this section, which extends [7], we present some of the most relevant contributions related to our proposal comparing them according to the requirements defined in Section 2.

Regarding the conceptual design and implementation of IoT, [13] proposes a UML profile for IoT physical devices considering fog and cloud concepts for objects interoperability and reusability. In the same way, [34] provides an automatic implementation framework for modeling different IoT systems using simple drag-and-drop designs. However, these works focus strictly on abstracting and solving (highly relevant) issues for the physical implementation of IoT rather than on data definition or integration. Other works have successfully integrated IoT data into different systems. For instance, [37] provides additional semantics to the design of wireless sensors networks to ease the further use of generated data. [26] defines a conceptual model for integrating IoT data into digital twins. Besides, [2] provides a meta-model for the integration of IoT data into web services.

In the section, we present existing works about designing complex and IoT data. [40] and [41] provide a survey of existing conceptual models for sensors and IoT data. They state that no existing work defines a data-centric model for data issued from sensors and IoT devices, respectively. So the authors propose a UML profile for modeling IoT data, which makes transparent all technical details related to the implementation. Moreover, [39] extends [40] to integrate sensors data with Stream Data Warehouses. However, these works do not support volatile data in the form of continuous queries, polyglot data associations are not possible, and non-functional requirements. Consequently, even through multiple approaches for building IoT architectures have been proposed, to the best of our knowledge, these approaches fail to provide a comprehensive formalism for representing volatile and persistent data while hiding the complex technical issues of their implementations.

In the context of database systems, numerous modeling methods (based on ER or UML) have been contributed for standard and temporal databases. For example, [11], proposes the usage of UML to represent temporal data properties, while [55] details an extension of the ER model. In the same way, some studies propose conceptual models for Data Warehouses, which can be considered as a particular kind of persistent data, through

UML [10] or ER extensions [12]. However, all these works do not take into account volatile data.

Conceptual models for stream data have not received much attention from the research community so far. To the best of our knowledge, only [9] proposed a UML profile for representing continuous queries integrated into a data warehouse approach. However, sensors data cannot be modeled nor non-functional requirements.

Numerous works studied the conceptual representation of Extraction, Transformation, Loading (ETL) processes. Extraction and loading are the most frequently researched operators (tasks). [50] and [3] proposed formal representations of the operators. They are similar to our *Input* and *Store* associations stereotypes. ETL models based on formal process notations, such as BPMN or UML Activity diagrams, are not adapted for our data-centric approach (as reviewed in [3]) since they model data and operations into separate diagrams. Only [50] adopted a data-centric approach. However, the approach is applicable to modeling only relational systems, and it ignores QoS notions, such as the *Latency* constraint, which we found necessary to model streaming-data applications as in IoT.

Non-Functional Requirements (NFR) can be defined as constraints attributes to define system quality and how it should perform. They are considered as one of the main aspects for the success of a system, because they provide an enormous help to understand at an early stage various problems of the system implementation. According to [21], NFR are goals to be achieved in the design pattern. Not taking into account NFR may generate more risks than functional requirements (FR) [20]. Nowadays NFR are more and more demanded by stakeholders, but they are mostly neglected or poorly handled. This affects the stakeholders decision making process, thus understanding and integrating NFR in system design can lead to better user experience and cost reduction.
In this context, several works have tried to give more importance to NFR, for instance [51] introduces an architecture with a process to separate, identify and integrate NFR, and [15] proposes a UML extension to define NFR and integrate them into different UML diagrams. However those works do not address IoT data.

In the context of IoT, to the best of our knowledge the majority of artifacts focus on the reliability and usability of the solution, but other NFR are not well taken into account [25]. [45] presents an agile approach to handle NFR such as (security and performance) in scrum, and [47] presents a NFR template and shows that NFR (cost, sensitivity, design complexity, storage, development process, environmental impact) can be very helpful to enhance IoT systems. [30] provides a UML-based approach to represent a variety of NFR in telecommunication domains, [8] proposes an MDA approach to handle the energy consumption of wireless sensor network using SysML and Modelica languages. However, these works do not focus on data design, and they do not take into account data and network NFR at the same time.

In the context of database systems, [36] shows that most of database design do not address NFR, and that in the future of database performance era, more NFR should be taken into consideration. Some works insist on the importance of NFR in the conceptual design [16],[14]. Indeed, some works propose to integrate NFR of the design process, such as [19] that propose an approach based on MDA and NFR integration to build database design, and [35] that details an approach with five steps to take NFR in consideration

before suggesting the appropriate database design (conceptual model, FR definition, NFR definitions, model fragmentation, database model).

[16] propose an ER framework that integrates NFR (validation, delivery,reliability and authorization) in the conceptual data models. In the same way, [27] propose to enrich the ER model with the workload of each entity in order to automatically generate the best NoSQL logical schema. These contributions remain focused on static persistent data, ignoring the particular features and challenges of volatile data, and network NFR.

Some works present NFR for real-time databases. [24] provides a UML profile for real time database modeling with temporal constraints and data quality, and [18] presents a UML package for real time objects with temporal constraints. Those works focus on temporally-constrained data, which is similar to our volatile data. Nonetheless, they disregard the existence of multiple subsystems that rely on particular technologies (e.g., sensors as the data generators) and the explicit association of different kinds of data.

Table 1 provides a summary and a comparison of most important works based on a data-centric approach previously described according to the requirements we have defined in Section 2: (i) Integrated design and implementation, which means the usage of a formal framework that can be used for representing also dynamic aspects of the system, (ii) The non-functional requirements for data and network, (iii) Types of data supported.

Table 1. Comparison of related work

Work	Integrated design and implementation	Non-Functional Requirements	Data Types
[40]	Yes	No	Partial (sensors data)
[41]	Yes	No	Partial (IoT devices data)
[39]	Yes	No	Partial (sensors data and stream data warehouse)
[11]	Yes	No	Partial (dictionary data)
[55]	No (ER formalism)	No	Partial (dictionary data)
[9]	Yes	No	Partial (not sensors data)
[16]	No (ER formalism)	Data (Dynamic - such as Usability, and Static - such as Accuracy)	Partially (dictionary data)
[27]	No (ER formalism)	Data (such as Volume)	Partially (dictionary data)
[24]	Yes	Data (such as Temporality)	Partial(sensors and dictionary data)
[18]	Yes	Data (Temporality)	Partial (dictionary and stream data)
Our Approach	Yes	Data (Temporality and Performance), Network (Latency)	All (persistent and volatile data)

From table 1 that reports only data-centric proposals, we can notice that all the works focus on NFR that concern data, and the majority of works only target few data types, and some of them are based on the ER formalism, which does not allow to represent dynamic aspects.

To conclude, existing works do not provide a unique and global conceptual representation of all involved data for complex IoT-based applications.

7. Conclusion and Future Work

IoT technologies are more and more used in all application domains, such as urban, health, tourism, etc. IoT provides decision-makers with complex real-time data at different spatio-temporal scales.

However, the adoption and deployment of IoT technologies raises a challenging research agenda related to standardizing design artifacts for IoT, sketching scalable architectures, and devising new algorithms for efficiently managing and processing IoT data at different levels. In particular, in the agriculture and agroecology context, IoT projects feature complex requirements, involving both heterogeneous hardware systems (e.g., robots, sensors, networks' hardware, and protocols, both in-situ and cloud servers), heterogeneous software systems, and complex spatio-temporal data. This complexity makes the design of conceptual data models of agroecology IoT applications mandatory for successful projects. The modeling has to encompass: (1) all heterogeneous data involved (i.e., from streamed sensors data, spatio-temporal data, to classical static and computed data) and (2) communication networks features.

Therefore, motivated by the lack of such a comprehensive conceptual framework, in [7], we proposed a UML profile to design data-centric agroecology IoT applications. We applied the UML profile for the monitoring of autonomous agricultural robots. Apart from feasibility implementation of the UML profile in a Big Data architecture, [7] does not present any validation of the proposed approach. Therefore, in this work, we provide a theoretical assessment of the UML profile based on existing metrics. Our experiments show the efficacy of our UML proposal from a theoretical point of view. Moreover, we provide a design and implementation methodology based on our approach, that can be integrated in classical existing software engineering methodologies. In order to validate this feature, we have shown by means of our real case study, how the UML profile class diagrams can be transparently used by UML sequence and activity diagrams.

Our UML profile does not allow representing multimedia data (video and images) that are commonly used in agriculture applications. Therefore, we plan to extend our profile to also represent multimedia data. Finally, setting the optimal (in practice sub-optimal) QoS values is challenging and it is considered a difficult optimization problem. A promising approach to supporting parameters and performance tuning is based on machine learning (ML) algorithms, e.g., [22,23,42]. Such algorithms require large volumes of test data to learn reliable performance models. Thus, excessive experimental evaluations are needed to provide performance data, to feed ML algorithms. In our project, tuning the parameters will be based on excessive experiments, therefore, we will address this issue in future work.

Acknowledgments. This work is supported by the French National Research Agency projects ANR-19-LCV2-0011 *Tiara*, and French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25).

References

1. Al-Sarawi, S., Anbar, M., Alieyan, K., Alzubaidi, M.: Internet of things (iot) communication protocols: Review. In: Proceedings of the 8th International Conference on Information Technology (ICIT). pp. 685–690. IEEE, Amman, Jordan (2017)
2. Alulema, D., Criado, J., Iribarne, L., Fernández-García, A.J., Ayala, R.: A model-driven engineering approach for the service integration of iot systems. Cluster Computing 23, 1937–1954 (2020)
3. Awiti, J., Vaisman, A.A., Zimányi, E.: Design and implementation of ETL processes using BPMN and relational algebra. Data & Knowledge Engineering 129, 101837 (2020)

4. Ayaz, M., Ammad-Uddin, M., Sharif, Z., Mansour, A., Aggoune, E.M.: Internet-of-things (IoT)-based smart agriculture: Toward making the fields talk. IEEE Access 7, 129551–129583 (2019)
5. Baccelli, E., Gündoğan, C., Hahm, O., Kietzmann, P., Lenders, M.S., Petersen, H., Schleiser, K., Schmidt, T.C., Wählisch, M.: Riot: An open source operating system for low-end embedded devices in the iot. IEEE Internet Things Journal 5(6), 4428–4440 (2018)
6. Basciani, F., Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: A tool-supported approach for assessing the quality of modeling artifacts. Journal of Computer Languages 51, 173–192 (2019)
7. Belhassena, A., Bimonte, S., Battistoni, P., Cariou, C., Chalhoub, G., Corrales, J.C., Laneurit, J., Moussa, R., Plazas, J.E., Wrembel, R., Sebillo, M.: On modeling data for iot agroecology applications by means of a UML profile. In: Proceedings of the 13th International Conference on Management of Digital EcoSystems. pp. 120–128. ACM, Virtual Event,Tunisia (2021)
8. Berrani, S., Hammad, A., Mountassir, H.: Mapping sysml to modelica to validate wireless sensor networks non-functional requirements. In: Proceedings of the 11th International Symposium on Programming and Systems (ISPS). pp. 177 – 186. IEEE, Algiers, Algeria (2013)
9. Bimonte, S., Schneider, M., Boussaid, O.: Business intelligence indicators: Types, models and implementation. International Journal of Data Warehousing and Mining 12(4), 75–98 (2016)
10. Boulil, K., Bimonte, S., Pinet, F.: Conceptual model for spatial data cubes: A UML profile and its automatic implementation. Computer Standards & Interfaces 38, 113–132 (2015)
11. Cabot, J., Olivé, A., Teniente, E.: Representing temporal information in UML. In: Proceedings of the 6th International Conference The Unified Modeling Language, Modeling Languages and Applications. pp. 44–59. Springer, San Francisco, CA, USA (2003)
12. Combi, C., Oliboni, B., Pozzi, G., Sabaini, A., Zimányi, E.: Enabling instant- and interval-based semantics in multidimensional data models: the t+multidim model. Information Sciences 518, 413–435 (2020)
13. Costa, B., Pires, P.F., Delicato, F.C.: Towards the adoption of omg standards in the development of soa-based iot systems. Journal of Systems and Software 169, 110720 (2020)
14. Cysneiros, L.M., Julio Cesar, S.d.P.L.: Integrating non-functional requirements into data modeling. In: Proceedings of the 4th International Symposium on Requirements Engineering. pp. 162–171. IEEE Computer Society, Limerick, Ireland (1999)
15. Cysneiros, L.M., Julio Cesar, S.d.P.L.: Non functional requirements: From elicitation to conceptual models. IEEE Transactions On Software Engineering 30(5), 328–350 (2004)
16. Cysneiros, L.M., do Prado Leite, J.C.S., de Melo Sabat Neto, J.: A framework for integrating non functional requirements into conceptual models. Requirements Engineering 6(2), 97–115 (2001)
17. Dalgaard, T., Hutchings, N., Porter, J.: Agroecology, scaling and interdisciplinarity. Agriculture, Ecosystems & Environment 100(1), 39–51 (2003)
18. DiPippo, L.C., Ma, L.: A uml package for specifying real-time objects. Computer Standards & Interfaces 22(5), 307–321 (2000)
19. Dubielewicz, I., Hnatkowska, B., Huzar, Z., Tuzinkiewicz, L.: Feasibility analysis of mda-based database design. In: Proceeding of the International Conference on Dependability of Computer Systems. pp. 19–26. IEEE Computer Society, Szklarska Poreba, Poland (2006)
20. Ebert, C.: Putting requirement management into praxis:dealing with nonfunctional requirements. Information and Software Technology 40(3), 175–185 (1998)
21. Gross, D., Yu, E.: From non-functional requirements to design through patterns. Requirement engineering 6(1), 18–36 (2001)
22. Hernández, Á.B., Pérez, M.S., Gupta, S., Muntés-Mulero, V.: Using machine learning to optimize parallelism in big data applications. Future Generation Computer Systems 86, 1076–1092 (2018)

23. Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F.B., Babu, S.: Starfish: A self-tuning system for big data analytics. In: Proceedings of the Fifth Biennial Conference on Innovative Data Systems Research. pp. 261–272. www.cidrdb.org, Asilomar, CA, USA (2011)
24. Idoudi, N., Duvallet, C., Bouaziz, R., Sadeg, B., Gargouri, F.: How to model a real-time database? In: Proceedings of the IEEE International Symposium on Object Component Service-Oriented Real-Time Distributed Computing. pp. 321–325. IEEE, Tokyo, Japan (2009)
25. Joseane, O.V.P., Rossana, M.C.A., Rainara, M.C.: Evaluation of non-functional requirements for iot applications. In: Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS. pp. 111–119. SCITEPRESS, Virtual event (2021)
26. Kirchhof, J.C., Michael, J., Rumpe, B., Varga, S., Wortmann, A.: Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. In: Proceedings of the 23rd International Conference on Model Driven Engineering Languages and Systems. pp. 90–101. ACM, Virtual event (2020)
27. Lima, C., Mello, R.S.: A workload-driven logical design approach for nosql document databases. In: Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services. pp. 73:1–73:10. ACM, Brussels, Belgium (2015)
28. Ma, Z., He, X., Liu, C.: Assessing the quality of metamodels. Frontiers of Computer Science 7(4), 558–570 (2013)
29. Marjani, M., Nasaruddin, F., Gani, A., Karim, A., Hashem, I.A.T., Siddiqa, A., Yaqoob, I.: Big IoT data analytics: Architecture, opportunities, and open research challenges. IEEE Access 5, 5247–5261 (2017)
30. Mehrdad, S., Cicchetti, A., Sjödin, M.: Uml-based modeling of non-functional requirements in telecommunication systems. In: Proceedings of the Sixth International Conference on Software Engineering Advances (ICSEA). pp. 213–220. The Institute of Electrical and Electronics Engineers, Barcelona, Spain (2011)
31. Melouk, M., Rhazali, Y., Hadi, Y.: An approach for transforming CIM to PIM up to PSM in MDA. In: Proceedings of the The 11th International Conference on Ambient Systems, Networks and Technologies. pp. 869–874. Elsevier, Warsaw, Poland (2020)
32. Muller, R.J.: Database design for smarties: using UML for data modeling. Morgan Kaufmann (1999)
33. Nathan Marz, J.W.: Big Data:Principles and best practices of scalable realtime data systems. Manning (2015)
34. Nepomuceno, T., Carneiro, T., Maia, P.H., Adnan, M., Nepomuceno, T., Martin, A.: Autoiot: a framework based on user-driven mde for generating iot applications. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC). pp. 719–728. ACM, Brno, Czech Republic (2020)
35. Noa, R., Shoval, p., Sturm, A.: A method for database model selection. In: Proceedings of the 20th International Conference, BPMDS Enterprise, Business-Process and Information Systems Modeling. pp. 261–275. Springer, Rome, Italy (2019)
36. Noa, R., Sturm, A.: Design methods for the new database era: a systematic literature review. Springer, Software Systems Modeling 19(2), 297–312 (2020)
37. Novacek, J., Kühlwein, A., Reiter, S., Viehl, A., Bringmann, O., Rosenstiel, W.: Lemons: Leveraging model-based techniques to enable non-intrusive semantic enrichment in wireless sensor networks. In: Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 561–568. IEEE, Portoroz, Slovenia (2020)
38. Omoniwa, B., Hussain, R., Javed, M.A., Bouk, S.H., Malik, S.A.: Fog/edge computing-based IoT (FECIoT): Architecture, applications, and research issues. IEEE Internet of Things Journal 6(3), 4118–4149 (2019)
39. Plazas, J.E., Bimonte, S., Corrales, M.S.J.C.: Self-service business intelligence over on-demand iot data: A new design methodology based on rapid prototyping. In: Proceedings of the New Trends in Databases and Information Systems (ADBIS). pp. 84–93. Springer, Lyon, France (2020)

40. Plazas, J.E., Bimonte, S., Schneider, M., de Vaulx, C., Battistoni, P., Sebillo, M., Corrales, J.C.: Sense, transform & send for the internet of things (sts4iot): Uml profile for data-centric iot applications. Data & Knowledge Engineering 139, 101971 (2022)
41. Plazas, J.E., Bimonte, S., de Vaulx, C., Schneider, M., Nguyen, Q., Chanet, J., Shi, H., Hou, K.M., Corrales, J.C.: A conceptual data model and its automatic implementation for iot-based business intelligence applications. IEEE Internet Things Journal 7(10), 10719–10732 (2020)
42. Popescu, A.D., Ercegovac, V., Balmin, A., Branco, M., Ailamaki, A.: Same queries, different data: Can we predict runtime performance? In: Proceedings of the ICDE Workshops. pp. 275–280. IEEE Computer Society, Arlington, VA, USA (2012)
43. Robinson, S., Arbez, G., Birta, L.G., Tolk, A., Wagner, G.: Conceptual modeling: definition, purpose and benefits. In: Proceedings of the Winter Simulation Conference. pp. 2812–2826. IEEE/ACM, Huntington Beach, CA, USA (2015)
44. Russom, P.: Data lakes: Purposes, practices, patterns, and platforms (2017), TDWI white paper
45. Sachdeva, V., Chung, L.: Handling non-functional requirements for big data and iot projects in scrum. In: Proceedings of the 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence. pp. 216–221. IEEE, Noida, India (2017)
46. Sapp, C.: Hyperscaling streaming analytics: Comparing stream analytics in the cloud with Amazon, IBM and Microsoft (2016), Gartner
47. Shubham, N.M., Keertikumar, B.M., Banakar, R.M.: Non functional requirement analysis in iot based smart traffic management system. In: Proceedings of the IEEE International Conference on Computing Communication Control and automation. IEEE, Pune, India (2016)
48. Souza, A.: Lambda architecture — how to build a big data pipeline (2019), https://dzone.com/articles/lambda-architecture-how-to-build-a-big-data-pipeli, dZone
49. Sørensen, C.G., Bochtis, D.: Conceptual model of fleet management in agriculture. Biosystems Engineering 105(1), 41–50 (2010)
50. Trujillo, J., Luján-Mora, S.: A UML based approach for modeling ETL processes in data warehouses. In: Proceedings of the Int. Conf. on Conceptual Modeling (ER). pp. 307–320. Springer, Chicago, IL, USA (2003)
51. Umar, M., Muhammad Naeem, A.K.: A framework to separate non-functional requirements for system maintainability. Kuwait Journal of Science Engineering (39), 211–231 (2012)
52. Vaisman, A.A., Zimányi, E.: Data Warehouse Systems - Design and Implementation. Springer (2014)
53. Wanner, J., Wissuchek, C., Janiesch, C.: Machine learning and complex event processing. A review of real-time data analytics for the industrial internet of things. Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 15, 1:1–1:27 (2020)
54. Zaharia, M., Ghodsi, A., Xin, R., Armbrust, M.: Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In: Proceedings of the Conf. on Innovative Data Systems Research, CIDR. www.cidrdb.org, Virtual event (2021)
55. Zimányi, E., Minout, M.: Implementing conceptual spatio-temporal schemas in object-relational dbmss. In: Proceedings of the On the Move to Meaningful Internet Systems: OTM Workshops. pp. 1648–1657. Springer, Montpellier, France (2006)
56. Zimanyi, E., Parent, C., Spaccapietra, S., Vangenot, C.: Multiple representation modeling. In: Liu, L., Ozsu, M.T. (eds.) Encyclopedia of Database Systems, 2nd Edition, p. 2419–2425. Springer (2018)

Sandro Bimonte is Research Director at French National Research Institute for Agriculture, Food and the Environment (France), and more exactly he is Member of the TSCF Laboratory. He received his PhD from INSA-Lyon, France (2004–2007). He is an Editorial Review Board Member of International Journal of Data Warehousing and Mining,

International Journal of Decision Support System Technology, and international conferences such as ER, DOLAP, etc. He has published more than 100 papers in refereed journals and international conferences. His research activities concern spatial data warehouses and spatial OLAP, visual languages, geographic information systems, spatio-temporal databases, geovisualisation, Big Data, and IoT. He joined and coordinated several research projects (such as VGI4bio.fr and BEYOND) on the above areas.

Hassan Badir is a full Professor in the Department of Computer Science and Engineering. He received a Ph.D. in Computer Science from the INSA-Lyon and Claude Bernard University (France) in 2005. He is actually: Head of Data engineering System TEAM SDET, Head of Bioinformatics and Data Science Master (BISD), Head of High Specialized Master study on Compute Engineering, Security and Decision, Co-Investigator of Human Heredity and Health African Bioinformatics Network H3ABionet, Head and founding member of the Moroccan Innovation and New trends of Information System Society. His research interests are in the areas of big data management, sensor and scientific data management, cloud computing and security, data management systems and distributed systems. I am also serving as an Associate Editor and Track Editor of International Journal of Database Management Systems(IJDBMS), Cluster Computing – The Journal of Networks, Software Tools and Applications (CLUSTE COMPUTING), International Society for Computers and Their Applications ISCA, Journal Concurrency and Computation: Practice and Experience. I am co-organiser of several workshops, programme chair of international conferences EGCM, ASD, EDA and INTIS. I am also serving as the scientific committee member of many reputed international conferences, AICSSA, MEDES, SITIS, EDA, ASD, EGC, INTIS, ASONAM, MedICT, SFC, DSC, ICCCI, BDCA, ICICS, JERI, BioMining, ICWIT, ICT-DM.

Pietro Battistoni is ending his PhD in Computer Science this year, 2021. He acquired an in depth understanding of software and hardware development project lifecycles by over 30 years of field research professional work experience in industrial automation. Twenty two years as owner of a privately held small company grant him experience in project management and technical skills in microcomputer-controlled system design, 3D CAD design, and electronic-CAD to develop hardware and firmware of electronic devices. Additional certified knowledge is in Cybersecurity, Cybersecurity Risk Management, Network Security, and Computer Forensics. His latest research interests are the IoT, distributed computing architectures and edge computing, geo-visual analytics, machine learning, human–computer interaction, robot programming, autonomous vehicles, and territorial intelligence, accomplished with 18 peer-review articles published in the lasts 2 years.

Houssam Bazza is a PhD student in computer science in Abdelmalek Essaadi University (Tangier-Morocco) and Membre of IDS Team, he held a master's degree in bioinformatics and data science from the national school of applied science morocco, his current research topics include: conceptual modeling, big data, business intelligence, stream data management and the internet of things.

Amina Belhassena is a Postdoctoral Researcher at Universite´ Clermont Auvergne, TSCF, INRAE, France. She received the PhD Degree in Computer Science from Harbin Institute

of Technology, China in 2018. She received the Master Degree of Technology in Computer Science from Abou Bakrbelkaid Tlemcen University, Algeria in 2012. Her research interest includes massive data computing, big data management, big streaming data management, data indexing and data mining.

Christophe Cariou is Research Engineer at INRAE, the French National Research Institute for Agriculture, Food and Environment. He received his Electrical Engineer Diploma in 1994, his Postgraduate Diploma in Electronics and Systems in 1995 and his PhD in 2012 on the control of poly-articulated robots. His research interests include the development of nonlinear, adaptive and predictive control for autonomous off-road mobile robots, as well as the optimal trajectory planning.

Gerard Chalhoub is an Associate Professor at University of Clermont Auvergne, France. He obtained his PhD Degree in Computer Science in 2009 at University of Blaise Pascal, France. He obtained his Habilitation Degree in Networking and Telecommunications in 2016 at University of Auvergne, France. His main research topics are reliability, quality of service and security in wireless networks. He has more than 70 publications in these domains.

Juan Carlos Corrales received the Dipl-Ing and master's degrees in telematics engineering from the University of Cauca, Colombia, in 1999 and 2004 respectively, and the Ph.D. degree in sciences, speciality computer science, from the University of Versailles Saint-Quentin-en-Yve-lines, France, in 2008. At present, he is a full professor and leads the Telematics Engineering Group (GIT) at the University of Cauca. His research interests focus on machine learning and data analytics.

Adrian Couvent is a Young Engineer and PhD Student in Robotics Applied to Agriculture. He is involved in robotics in several aspects (from control laws to user experience). His thesis on taking into account the competence of an operator in adjusting the degree of autonomy of a technical system will be defended in early 2022.

Jean Laneurit obtained his PhD at Institute Pascal, France in 2006. He is currently member of TSCF, INRAE, Clermont Ferrand. He is Research Engineering in Computer Science Applied to Robotics.

Rim Moussa is an Associate Professor at University of Carthage. She received her MSc and PhD in Distributed Databases from Universite´ Paris IX Dauphine (France). Her current research interests include scalable and distributed data management systems, Big Data architectures and spatial computing at scale.

Julian Eduardo Plazas is a Ph.D. in Telematics Engineering and Computer Science in Universidad del Cauca (Colombia) and Université Clermont Auvergne (France). He has worked in the definition of rural Early Warning Systems through Converged Services and Complex Event Processing for agricultural environments, in the implementation of Machine-Learning Classifiers for agricultural Decision Support Systems, and in Conceptual Modelling for agricultural Wireless Sensor Networks. His current research topics include the Internet of Things, Business Intelligence, Data Analytics and Conceptual Modelling for Intelligent Agriculture Systems.

Monica Sebillo received a Laurea Degree in Scienze dell'Informazione from the University of Salerno and a PhD in Applied Mathematics and Computer Science from the University of Naples. She is an Associate Professor in Computer Science at the Department of Computer Science (DI) at the University of Salerno. Monica is an ACM Senior Member. Her research interests include geographic information systems, GeoAI and geospatial databases.

Nicolas Tricot is currently Research Fellow at INRAE (National Research Institute for Agriculture, Food and the Environment) in the Research Unit TSCF (Technologies and Information Systems) at Clermont-Ferrand (France). He received an Engineer Degree and a Master Degree in Automation in 2001. He defended is PhD in 2005 on the Topic of Design and Evaluation of Advanced Driving Assistance Systems. He joined Irstea in 2006. In a first time, he worked on the integration of human factors in system design. In a second time, he joined the research team Romea (Robotic and Mobility for Environment and Agriculture) in ClermontFerrand to work on human and agricultural robot interactions.

Received: March 01, 2022; Accepted: August 22, 2022.