information

MDPI

Article

Configurable Distributed Data Management for the Internet of the Things †

Nikos Kefalakis, Aikaterini Roukounaki and John Soldatos *

Athens Information Technology, 15125 Athens, Greece; nkef@ait.gr (N.K.); arou@ait.edu.gr (A.R.)

* Correspondence: jsol@ait.gr; Tel.: +302106682700

† This is an extended and updated version of our paper presented in 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini Island, Greece.

Received: 9 October 2019; Accepted: 16 November 2019; Published: 20 November 2019

Abstract: One of the main challenges in modern Internet of Things (IoT) systems is the efficient collection, routing and management of data streams from heterogeneous sources, including sources with high ingestion rates. Despite the existence of various IoT data streaming frameworks, there is still no easy way for collecting and routing IoT streams in efficient and configurable ways that are easy to be implemented and deployed in realistic environments. In this paper, we introduce a programmable engine for Distributed Data Analytics (DDA), which eases the task of collecting IoT streams from different sources and accordingly, routing them to appropriate consumers. The engine provides also the means for preprocessing and analysis of data streams, which are two of the most important tasks in Big Data analytics applications. At the heart of the engine lies a Domain Specific Language (DSL) that enables the zero-programming definition of data routing and preprocessing tasks. This DSL is outlined in the paper, along with the middleware that supports its runtime execution. As part of the paper, we present the architecture of the engine, as well as the digital models that it uses for modelling data streams in the digital world. We also discuss the validation of the DDA in several data intensive IoT use cases in industrial environments, including use cases in pilot productions lines and in several real-life manufacturing environments. The latter manifest the configurability, programmability and flexibility of the DDA engine, as well as its ability to support practical applications.

Keywords: distributed data analytics; big data; industrial Internet of Things; industry 4.0

1. Introduction

We are currently witnessing the rise of the Internet of Things (IoT) paradigm as a result of the proliferation of the number and type of internet-connected devices. IoT deployments with the highest business value are in most cases found in the scope of industrial environments and are conveniently called Industrial IoT (IIoT) deployments. IIoT is the cornerstone of the fourth industrial revolution (Industry 4.0), which is characterized by the digitization of physical processes in industrial environments such as manufacturing shop floors, energy plants and oil refineries. The latter digitization is primarily based on the deployment of Cyber Physical Systems (CPS), such as sensors, automation devices and smart objects like drones and automated guided vehicles.

The vast majority of IIoT use cases involve collection and processing of data from a variety of distributed data sources, including processing and analytics over data streams with very high ingestion rates. As a prominent example, predictive maintenance applications involve the application of machine learning and artificial intelligence algorithms over multisensory data (e.g., vibration, temperature, ultrasonic, thermal imaging) towards predicting the end of life of machines and equipment. Likewise, Zero Defect Manufacturing (ZDM) applications analyze large amounts of digital data from various automation devices in order to proactively identify the causes of defects. As another example, digital

twin applications apply advanced analytics over real and simulated data sources in order to experiment with what-if scenarios and take optimal production planning and industrial automation decisions. Therefore, Distributed Data Analytics (DDA) infrastructures are at the heart of IIoT systems and applications, which must deal with BigData problems, notably problems that analyze very large numbers of distributed and heterogeneous data streams that usually exhibit high velocity as well.

In this context, DDA infrastructures for IIoT applications must provide support for:

- High-performance and low-latency operations, i.e., low overhead, near real-time processing of data streams.
- Scalability, i.e., ability to process an arbitrarily large number of streams, with only marginal upgrades to the underlying hardware infrastructure.
- Dynamicity, i.e., ability to dynamically update the results of the analytics functions, upon changes in their configuration, which is essential in cases of in volatile industrial environments where data sources may join or leave dynamically.
- Stream handling, i.e., effective processing of streaming data in addition to transactional static or semi-static data (i.e., data at rest).
- Configurability, i.e., flexible adaptation to different business and factory automation requirements, such as the calculation of various KPIs (Key Performance Indicators) for production processes involving various data sources on diverse industrial environments that may even leverage different streaming middleware platforms and toolkits.

The importance of streaming analytics for IoT and BigData applications has given rise to several research and industry efforts towards producing high performance stream processing engines. A first generation of streaming analytics applications used centralized stream processing engines such as Aurora [1] and TelegraphCQ [2]. These engines provided window-based query operators that execute continuous queries over relational data streams. They supported the relational query model (e.g., such as the CQL (Continuous Query Language) [3]) but lacked support for parallel data processing. The increase of stream rates and query complexity drove another generation of stream processing engines, which were distributed and could harness the processing power of a cluster of stream processors. Typical examples of such systems are Borealis [4] and InfoSphere Streams [5], which permit parallelism for continuous queries i.e., one query can be executed on multiple machines. Such systems exploit task parallelism, i.e., they execute different operators on different machines and allow the execution of many different continuous queries in parallel.

Systems like InfoSphere Streams support intraquery parallelism, which specifies stream connections, but management and configuration is manual. This poses limitations for BigData applications, where a single continuous query must in several cases process a large volume of streaming data. To alleviate this limitation, stream processing engines that focus on intraquery parallelism emerged. The latter parallelize the execution of individual query operations. Typical examples include StreamCloud [6] and the popular Apache S4 and Apache Storm systems, which express queries as directed acyclic graphs with parallel operators interconnect by data streams. However, these systems cannot scale out the computation at runtime and therefore are not effective in supporting unknown BigData analytics jobs when the computational resources needed are not known ahead of time. To this end, systems like Spark Streaming [7] that parallelize streaming queries by running them on the Spark-distributed dataflow framework using microbatching have emerged. Microbatching permits the execution of streaming computations as a series of short-running Spark jobs that output incremental results based on the most recent input data. Nevertheless, these execution models have limitations when it comes to supporting sliding windows in the streaming process. In recent years, the Apache Flink engine [8] incorporated a distributed dataflow framework that can execute data-parallel batch and streaming processing jobs on the same platform. Computation is described as dataflow graphs, which are optimized using existing database techniques. This is the reason why Apache Flink and Apache Spark are currently two of the most popular streaming engines

for Big Data systems. However, these platforms assume that stream processing operators are stateless. While this simplifies scalability and failure recovery, it is a setback to expressing complex analytic tasks such as data mining and machine learning algorithms that need to refine a model incrementally. To address this problem, some of the state-of-the-art stream processing engines adopt a stateful stream processing model. This is for example the case with Apache Samza [9] and Naiad [10], which execute streaming operators in a data-parallel fashion. More recent streaming engines implement the concept of Stateful Dataflow Graphs (SDGs), which are graphs containing vertices that are data-parallel stream processing operators with arbitrary amounts of mutable in-memory state, and edges that represent the stream. SDGs can be executed in a pipelined fashion and have a low processing latency. When processing SDGs, the machines take checkpoints of the in-memory state of processing operators in a cluster, which are persisted to disk. Therefore, in case a machine in the cluster fails, the failed operator instances can be restored on other machines, recovering their state from the most recent checkpoint, and reprocessing buffered stream data in order to restore the operator's state in a way that ensures it is up to date.

Recently, IIoT vendors and solution integrators make also extensive use of the Kafka Streams framework [11], which incorporates many of the previous listed functionalities and addresses many of the stream processing challenges in IIoT environments. It supports event-at-a-time processing with millisecond latency, which alleviates the limitations of microbatching, while at the same time, providing the means for stateful processing including distributed joins and aggregations. Moreover, it supports distributed processing and failover, while at the same time offering a convenient DSL (Domain Specific Language) for defining queries.

The configurability functionalities of state-of-the-art streaming engines fall short when it comes to routing data streams across the different data producers and data consumers of IIoT environments. Indeed, most streaming engines are configured based on low-level APIs (Application Programming Interfaces), which incur significant programming effort. Moreover, zero-programming options functionalities like the Kafka Streams DSL are very versatile when there is a need to dynamically select data sources from the shop floor and to define queries over them. Nevertheless, they are not very effective when there is a need to define how data should be routed across many different data consumers (e.g., predictive analytics algorithms, digital twins) in industrial environments. In other words, they are very good for defining low-level tasks (e.g., configuration of window sizes and stateful graphs), but do not make provisions for the preprocessing and routing of the data across different producers and consumers. Likewise, they are also very much tailored to clustered environments, yet they provide no ready to use facilities for configuring and deploying streaming applications in edge computing environments, which are very common in the case of IIoT [12]. Specifically, the concepts of edge/fog nodes and how to collect and analyze data across them are not directly supported in the configuration capabilities of state-of-the-art streaming engines.

Most important, it is quite common for IIoT environments to deploy different streaming engines and toolkits e.g., Spark, Kafka and Storm can be used for different applications running in different parts of the plant. In such cases, the need for routing data streams from different streaming middleware platforms to consumers and applications is likely to arise. Hence, there is a need for a meta-streaming engine for streaming data routing and processing, which can make sure that data streams acquired by different streaming engines are delivered in the target application. Consider for example a predictive maintenance application, which should typically combine data streams from multiple sensor-based applications, along with data streams and data at rest from business information systems (e.g., quality data from an Enterprise Resource Planning (ERP) system). The concept of such a meta-level streaming engine is perfectly in-line with edge computing systems, as the latter are likely to combine different streaming middleware platforms in their edge nodes.

Meta-level streaming functionalities can be defined by means of a domain specific language [13], which provides versatility in configuring domain-specific data operations in ways that ease programming and lower development times [14]. There are various DSL languages for IoT systems,

which focus however on the needs of different types of systems and applications like RFID and multisensory applications for supply chain management (e.g., [15,16]). Moreover, there are various approaches to designing DSL, which vary in terms of efficiency [17,18]. Nevertheless, existing DSL do not adequately cover the domain of routing and preprocessing of heterogeneous streams for IIoT applications, which is one of the main motivations behind the work presented in this manuscript. Note also that DSLs provide a foundation for almost zero-programming applications in the domains that they target, since they can be coupled with programming environments and can be amenable by visual tools [19]. In this context, the DSL that is presented in the paper lowers the effort required for development of IIoT data analytics applications and aligns with model driven engineering concepts and principles [20,21].

In the remaining of the paper we introduce a configurable infrastructure for distributed data analytics environments, which caters for the flexible and almost zero-programming configuration of data routing and preprocessing functions. The presented DDA (i.e., the FAR-EDGE DDA) is a meta-level streaming engine that provides the means for routing and preprocessing data streams regardless of the streaming middleware/engine used to capture them. The FAR-EDGE DDA is configurable in the scope of edge computing environments, as it defines an Edge Analytics Engine (EAE) that handles analytics within the scope of an edge node, while at the same time providing mechanisms for combining and synchronizing analytics functionalities across multiple edge nodes. The latter functionalities are very common in IIoT environments, where several processes (e.g., production scheduling, supply chain management) are likely to involve multiple stations or entire plants i.e., multiple edge nodes using different streaming middleware toolkits (e.g., Kafka or Spark). The configurability of the FAR-EDGE DDA hinges on an abstract modelling of data sources and data processing functions, along with their structuring against edge nodes, edge analytics functions and analytics spanning multiple edge nodes. The management of the respective data models is what facilitates configurability. In principle, the FAR-EDGE DDA offers its own DSL for managing routing and processing of data streams from any streaming engine (e.g., Kafka, Spark, Storm) to any type of consuming application (e.g., rule engine, machine learning, artificial intelligence). This paper is a significantly extended version of a conference paper of the co-authors. It enhances the conference paper in three main directions:

- First, it introduces some of the foundational concepts of the DDA, such as the DSL for describing analytics pipelines. The latter DSL is also illustrated through a practical example.
- Second, it also introduces the concept of the Common Interoperability Registry (CIR) for linking data sources that are described based on other schemes, different that the project's DSLs. The CIR concept was not part of the conference paper.
- Third, it provides more details and richer information about the architecture of the DDA and its use in the scope of cloud/edge deployments.

The rest of paper is structured as follows: Section 2 introduces the architecture, main elements and principles of operation of the DDA. Section 3 present the underlying data models that enable its configurability, including flexible ways for ensuring the interoperability of different data sources and databases that contain data of interest to IIoT applications. Section 4 provides information on the open source implementation of the DDA, along with its deployment in several applications. The latter applications serve as a basis for the technical validation of the DDA. Finally, Section 5 is the concluding section of the paper.

2. Architecture of the Configurable Data Analytics Engine

2.1. DDA Overview

The FAR-EDGE DDA is a configurable infrastructure for Distributed Data Analytics, which enables the collection, aggregation, integration and preprocessing of diverse data streams. It is a distributed middleware infrastructure, which is agnostic of the underlying data sources and streaming

analytics middleware toolkits. As such, it provides the means for configuring and deploying streaming analytics applications over a variety of heterogeneous data sources, including streaming analytics engines such as Apache Spark and Kafka.

As outlined in Figure 1, the FAR-EDGE DDA is structured in tiers in line with most reference architectures for IoT applications, including an edge, a cloud and a ledger tier. The edge tier provides the means for low-level, real-time analytics operations that are executed close to the field, typically in the scope of a local area network. The FAR-EDGE DDA comprises an Edge Analytics Engine (EA-Engine), which is instantiated and executed at the edge of the network. On the other hand, the cloud tier enables the execution of analytics that span multiple edge nodes (and EA-Engine instances) by means of a cloud-based Distributed Analytics Engine (DA-Engine). Hence, the DA-Engine executes analytics functions over data streams that are aggregated in the cloud. It can handle larger amounts of data than the EA-Engine, yet is not suitable for real-time, low overhead stream processing. The FAR-EDGE DDA introduces also a ledger tier, which provides the means for decentralized configuration and synchronization of edge analytics processes, using a distributed database based on blockchain technology. The description of this novel blockchain-based configuration and synchronization is beyond the scope of this paper. Instead, interested readers can consult [22,23].

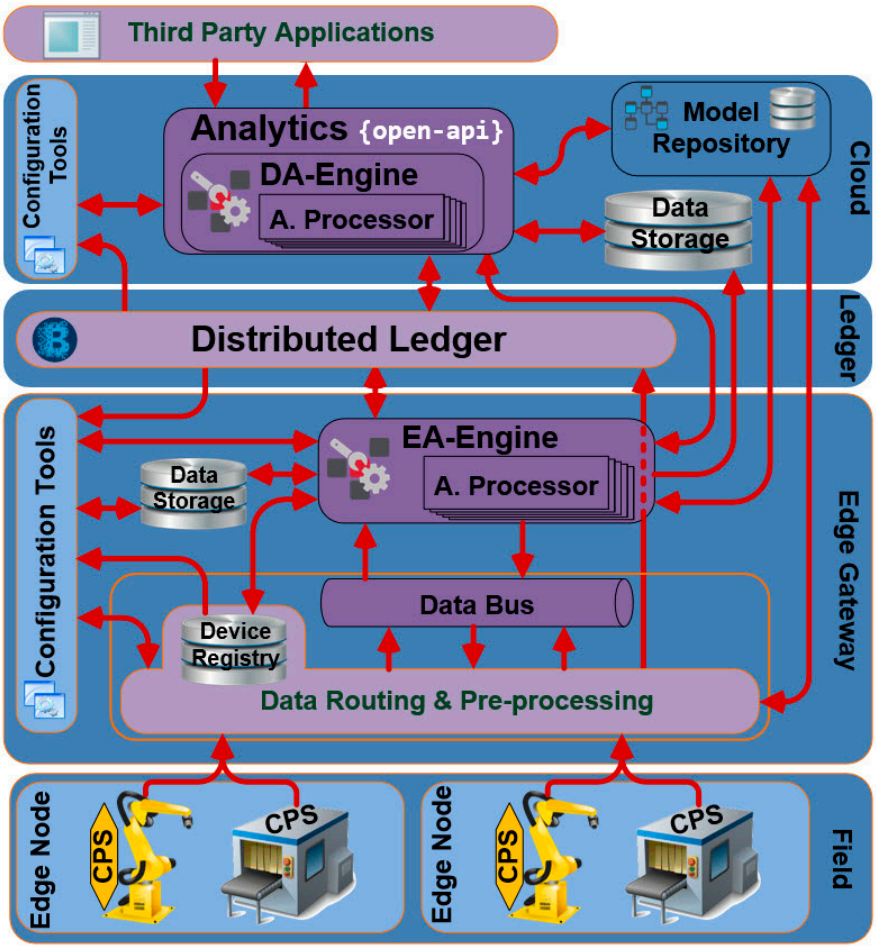


Figure 1. Anatomy of the Distributed Data Analytics Engine (see [16]).

2.2. Edge Analytics Engine (EA-Engine)

The EA-Engine is deployed and executed close to the field i.e., within an edge gateway. Its operation is based on a Data Routing and Preprocessing (DR&P) component, a data bus and a registry of devices (i.e., Device Registry). Specifically, the DR&P component routes data from the data sources (e.g., industrial platforms and devices that may include data streaming platforms) to the Edge Analytics

Engine (EA-Engine). The routing operations are based on information contained within the Device Registry, which contains information (e.g., connectivity protocol, IP address and port) about how the various devices and data sources can be accessed. The registry ensures the dynamism of the EA-Engine, data sources can at any time register or deregister from the device registry. Moreover, the component provides preprocessing capabilities, which allow for transformations to data streams prior to their delivery to the EA-Engine.

In addition to interacting with the Device Registry, the DR&P component provides a Data Bus, which is used to route streams from the various devices to appropriate consumers, i.e., processors of the EA-Engine. The Data Bus is not restricted to routing data streams stemming directly from the industrial devices and other shopfloor data sources. Rather it can also support the routing of any data streams and events that are produced by the EA-Engine. Overall, the EA-Engine is a configurable runtime environment hosted in an edge gateway, which executes data analytics close to the field in order to meet stringent latency requirements.

Towards configuring the operation of the EA-Engine, DDA application developers and solution providers can edit a specification of analytics tasks, which is provided as an XML (eXtensible Markup Language) file and serves as a DSL for customizing EA-Engine's operation. This DSL provides the means for expressing streaming analytics functions based on the combination of several processing functions that are conveniently called "processors". The latter processing functions operate over streaming data that are available in the Data Bus of the DDA.

The EA-Engine supports three different types of processors, namely:

- **Preprocessors**, which preprocess (e.g., filtering) data streams and prepare them for analysis by other processors. Preprocessors acquire streaming data through the DR&P component and produce new streams that are made accessible to other processors and applications through the Data Bus of the infrastructure.
- **Storage processors**, which store streams to some repository such as a data bus, a data store or a database. They provide the persistence functions, which is a key element of any data analytics pipeline.
- **Analytics processors**, which execute analytics processing functions over data streams ranging from simple statistical computations (e.g., average or a standard deviation) to more complex machine learning tasks (e.g., execution of a classification function). Like preprocessors, analytics processors can access and persist data to the Data Bus.

The EA-Engine can execute analytics pipelines comprising combinations of these three processors (as shown in Figure 2). The relevant pipelines (or workflows) are described through well-defined configuration files, called Analytics Manifests (AMs), which essentially represent the important part of the DSL of the EA-Engine. Specifically, an AM defines a set of analytics functionalities as a graph of processing functions that comprises the above three types of processors and which can be executed by the EA-Engine. AM instances are built based on the devices, data sources, edge gateways and analytics processors that are available. All these devices and data sources are models based on the digital modelling approach that is described in the following Section of the paper and which specifies additional constructs of the DSL of the DDA.

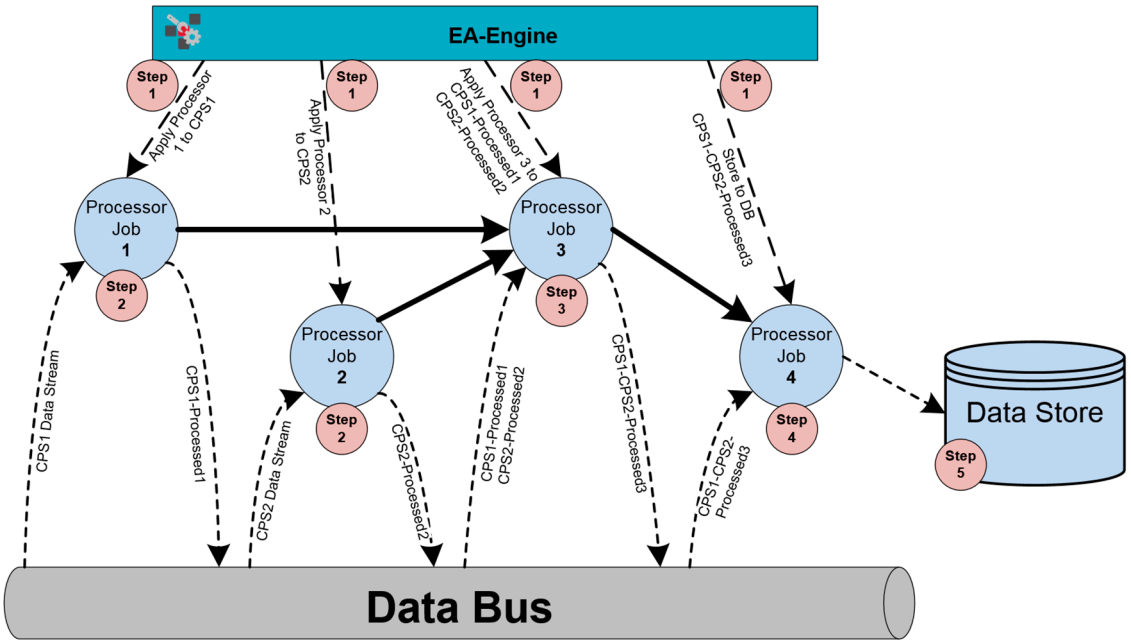


Figure 2. Configurable Data Preprocessing at the Edge.

An EA-Engine specifies and executes edge analytics pipelines that comprise combinations of the above-described processing functions. Factory wide analytics comprising multiple edge analytics workflows in a higher-level pipeline are then executed through the distributed analytics engine (DA-Engine) that resides in the cloud layer of a DDA deployment.

2.3. Distributed Analytics Engine (DA-Engine)

The DA-Engine is destined to execute global analytics functions based on analytics configurations that span and combine multiple edge analytics instances. It is also configurable and programmable thanks to its support for a DSL that describes global analytics functions in terms of edge nodes, edge gateways, data sources and the processing functions that are applied over them. In particular, the DSL is specified as an Analytics Manifest (AM) for global analytics, which can comprise multiple edge analytics instances specified as AMs as well. The DA-Engine leverages the descriptions of all these artifacts within a digital models' repository, which comprises the digital representation of the devices, data sources and edge gateways that are part of the DDA. The structure of these Digital Models is also described in the following section. Note that all the digital models are kept up to date and synchronized with the status of the DDA's elements. Hence, they are accessible from the DR&P, the EA-Engine and the DA-Engine components and can be used in the specification of both edge analytics and global analytics tasks. As already outlined, the DA-Engine stores data within a cloud-based data storage repository, which persists the results of global analytics tasks.

2.4. Open API for Analytics

The FAR-EDGE DDA infrastructure defines, implements and exposes an Open API (Application Programming Interface). This API enables external systems to access and integrate the functionalities of the DDA infrastructure, including the configuration, execution and deployment of factory-wide analytics tasks, that span multiple edge gateways. The Open API enables solution integrators to configure the DDA and to execute data processing and analytics functions over data streams stemming from all devices that registered in the registries of the DR&P components of the edge nodes of the DDA. In this way the DDA infrastructure can be used by third-party applications. In practice the Open API supports the following three types of operations:

- CRUD (Create Update and Delete) operations for AMs, which enable the management of AMs.

- Access to and management of information about deployed instances of analytics workflows, including access to information about AMs and their status.
- Management of a specified AM, including starting, stopping, posing and resuming the execution of an analytics instance expressed in the AM DSL.

The Open APIs have been specified and implemented as RESTful APIs, which facilitates their use by application developers and solution integrators.

3. Digital Modelling and Common Interoperability Registry

3.1. Overview

The configuration and execution of global analytics workflows is based on the management of appropriate digital models that specify the data sources, the data processing functions, as well as the edge gateways entailed in these workflows. The digital models provide a representation of such analytics workflows and enable the management of this representation (e.g., its execution of updates and deletion of operations). To this end, the digital models that empower the configuration of the DDA follow a hierarchical structure, which specifies the different relationships between the various entities. This structure specifies that an edge gateway comprises multiple data sources, each of which has its own definition and can be instantiated multiple times. Moreover, each instance produces data associated with the data sources. As an example, Figure 3 illustrates a snapshot of the digital models' structure, which shows the association of APM (Analytics Processor Manifest) with various data sources. The role of the APM is illustrated in the following paragraph.

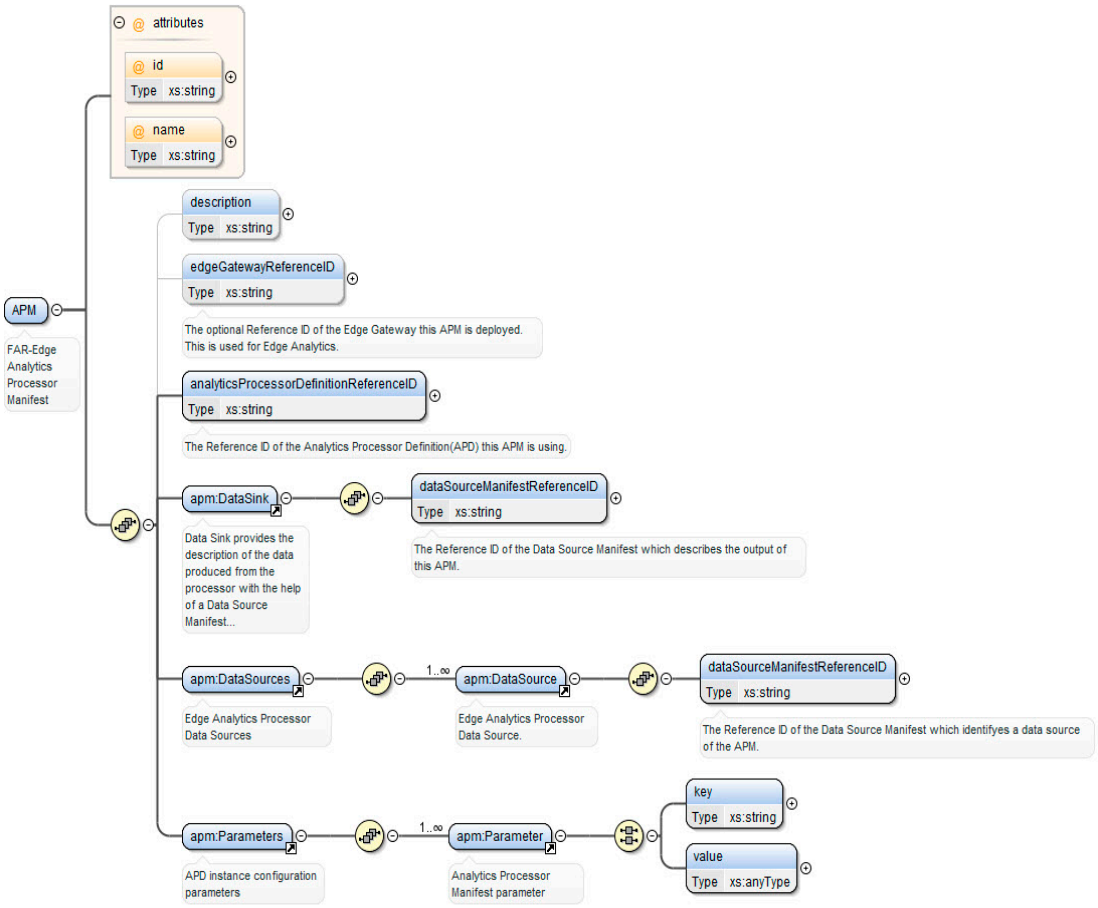


Figure 3. Elements of an APM Schema.

The specification of data sources, processors, edge gateways and other entities outlined above provides the means for configuring the DDA based on a basic set of metadata (e.g., location, types) about edge nodes and data sources. Nevertheless, IIoT analytics applications are likely to involve additional metadata residing in a variety of systems and databases. As a characteristic example, predictive maintenance systems can benefit from a variety of metadata from non-IoT systems (e.g., ERP and Quality Management systems) that are related to live data sources. To accommodate such metadata as part of an analytics process, the data models' infrastructure of the DDA engine provides the means for linking data sources and their data with metadata from other databases in an interoperable way. To this end, a Common Interoperability Registry (CIR) approach is implemented in order to enable the linking and interoperability of datasets that refer to the same physical or logical entity (e.g., a piece of equipment or an automation device). Following paragraphs illustrate the core digital models of the engine, as well as their extensibility with additional information as part of a CIR approach. Figure 4 illustrates the root of the Data Models and the subschemas that comprise them, which include subschemas for data definitions, observations and CIRs. Further paragraphs provide more detailed about the modelling of plant data and metadata (i.e., data definitions), as well as for the modelling of analytics pipelines in the DDA. Nevertheless, the complete specification of the FAR-EDGE data models is several hundreds of pages and beyond the scope of this manuscript. Interested readers can refer to the open source implementation of the DDA, which includes the relevant schemas and their documentation.

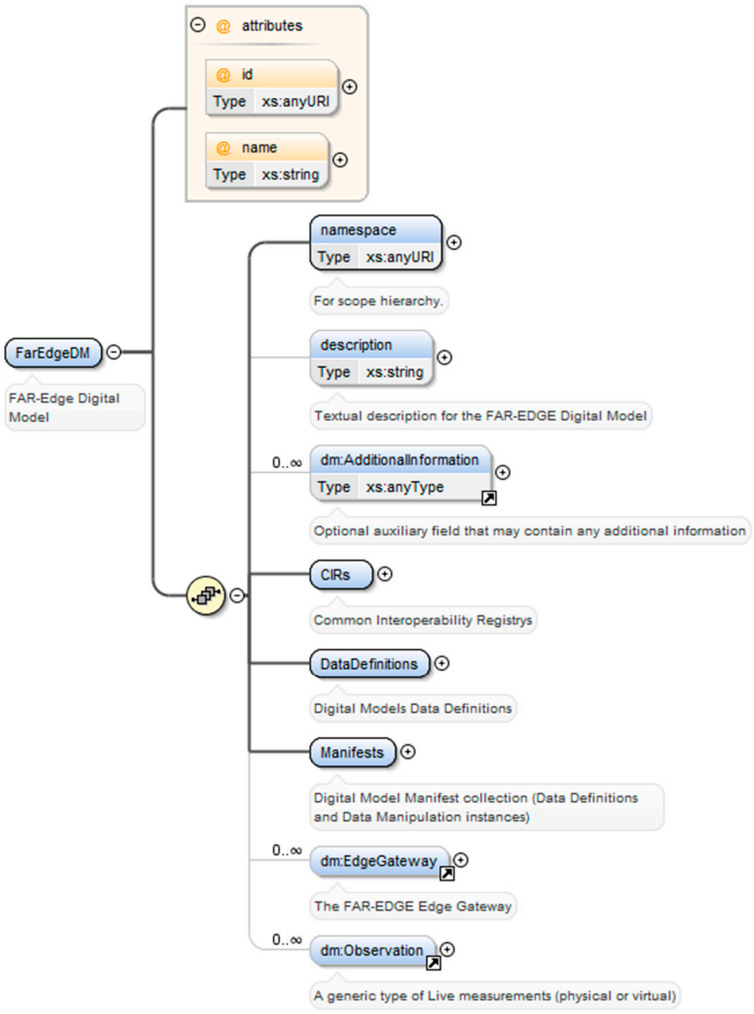


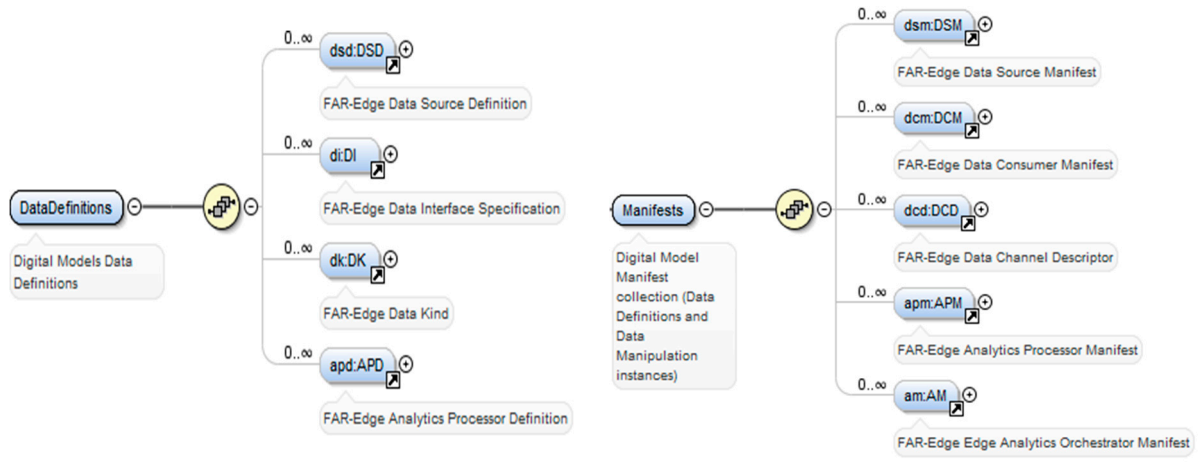
Figure 4. Root of the FAR-EDGE Data Models.

3.2. Plant Data and Metadata

Factory data and metadata are modeled based on the following entities (see also Figure 5):

- **Data Source Definition (DSD)**: Defines the properties of a data source in the shop floor, such as a data stream from a sensor or an automation device.
- **Data Interface Specification (DI)**: It is associated with a data source and provides the information needed to connect to it and access its data (e.g., network protocol, port, network address).
- **Data Kind (DK)**: This specifies the semantics of the data of the data source. It can be used to define virtually any type of data in an extensible way.
- **Data Source Manifest (DSM)**: Specifies a specific instance of a data source in line with its DSD, DI and DK specifications. Multiple manifests are therefore used to represent the data sources that are available in the factory.
- **Data Consumer Manifest (DCM)**: Models an instance of a data consumer, i.e., any application that accesses a data source.
- **Data Channel Descriptor (DCD)**: Models the association between an instance of a consumer and an instance of a data source. Keeps track of the established connections and associations between data sources and data consumers.
- **LiveDataSet**: Models the actual dataset that stems from an instance of a data source that is represented through a DSM. It is t is associated with a timestamp and keeps track of the location of the data source in case it is associated with a mobile edge node. In principle, the data source comprises a set of name–value pairs, which adhere to different data types in line with the DK of the DSM.
- **Edge Gateway**: Models an edge gateway of an edge computing deployment. Data sources are associated with an edge gateway, which usually implies not only a logical association, but also a physical association as well.

Based on the above entities, it is possible to represent the different data sources of a digital shopfloor in a modular, dynamic and extensible way. This is based on a repository (i.e., registry) of data sources and their manifests, which keeps track of the various data sources that register to it. The FAR-EDGE platform includes such a registry, which provides dynamicity in creating, registering and using data sources in the industrial plant [24].



(**a**) (**b**)

Figure 5. Data Definitions and Manifests Subschemas. (**a**) Factory Data Definitions and Manifests Subschemas (**b**) metadata Definitions and Manifests Subschemas.

3.3. Data Analytics Data and Metadata

Analytics workflows and pipelines are modelled based on the following entities:

- **Analytics Processor Definition (APD)**: Specifies processing functions applied on one or more data sources. As outlined, three types of processing functions are supported, including data preprocessing, data storage, and data analytics functions. These three types of functions can be combined in various configurations over the data sources in order to define analytics workflows.
- **Analytics Processor Manifest (APM)**: Represents an instance of a processor that is defined through an APD. Each instance specifies the type of processor and its actual logic through linking to an implementation function (like a Java class).
- **Analytics orchestrator Manifest (AM)**: Represents an analytics workflow as a combination of analytics processor instances (i.e., APMs). It is likely to span multiple edge gateways and to operate over their data sources.

3.4. Common Interoperability Registry

The above-listed digital models are sufficient for specifying and executing configurable distributed streaming analytics tasks. However, they only provide a simple set of metadata to application developers, such as the location of edge gateways and the timestamp of data sources. In IIoT applications, it is likely that streaming analytics are performed over a variety of data sources, beyond the ones that follow the format and semantics of the digital models as outlined above. In order to make use of such analytics data sources, the introduced engine implements an object identification registry, which aims at supporting both static and dynamic data sources in diverse IIoT analytics scenarios. The registry facilitates storage of observed information for a great variety of different data sources, through linking the observations with the objects that they concern in the data models. Hence, it also supports the integration and use of third-party databases as a means of enriching data observations with more semantics.

The implementation of the CIR is outlined in Figure 6 and was realized in the scope of the H2020 PROPHESY (www.prophesy.eu) as an add-on extension to the FAR-EDGE DDA infrastructure. It enables the linking of repositories that contain relevant information about the objects and data sources. The overall enhanced engine solution comprises:

- A database of the digital models presented in the previous paragraphs i.e., models representing diverse data sources such as sensors and automation devices deployed in the field.
- The CIR, which provides the infrastructure for linking objects' data that resides in different databases. In this way, the CIR enriches of the datasets of the core digital models with additional data and metadata residing in "linked" databases.

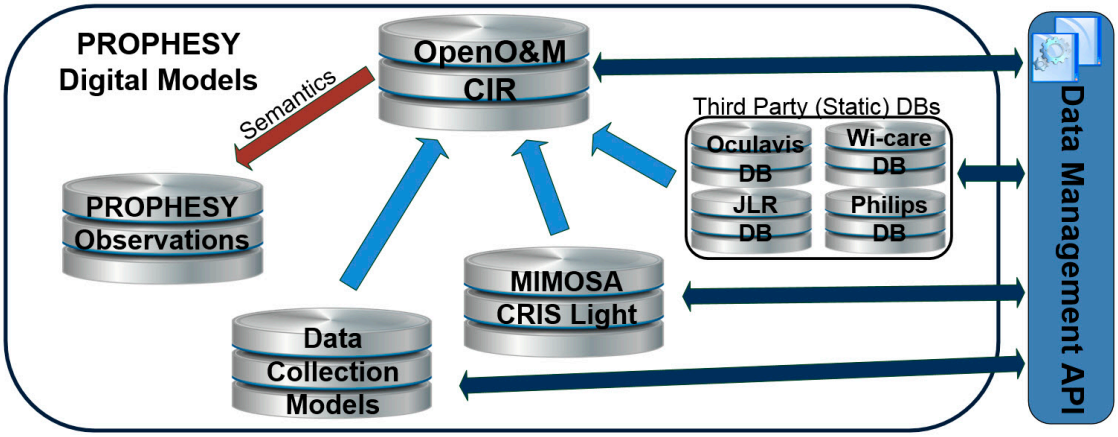


Figure 6. Concept of CIR Approach as implemented in the PROPHESY Project.

By enhancing the core data models with the CIR, application developers are offered the means to access consolidated datasets from all linked databases. They can access enhanced datasets along with proper metadata that describe the context of the objects (e.g., machines, sensors, automation devices) of the application.

Figure 4 illustrates how the CIR database links different databases for predictive maintenance, including: (i) The PROPHESY project database where maintenance data sources are modelled based on the constructs described in the previous paragraphs (e.g., constructs like DSD, DI and DK); (ii) A database containing maintenance datasets compliant to the MIMOSA Open Standards for Physical Asset Management [25] and more specifically to the MIMOSA CRIS (Common Relational Information Schema (CRIS) schema; (iii) Various proprietary third party databases maintained by different partners of the PROPHESY project consortium. The CIR infrastructure links the information about objects and entities that are referenced by more than one database, which enables analytics over enhanced datasets. Using the CIR, an application developer can access information about the full context and observations that relate to an object, regardless of the repository where these observations reside. Note also that the CIR implementation adheres to standards prescribed by the MIMOSA Open Standard for Operations and Maintenance (Open O&M), which ensures neutrality, openness and technological longevity. In practice, the CIR implements an XML schema and a relational database in line with the specifications and the open source libraries of the Open O&M that can be found in the MIMOSA organization's GitHub.

In addition to enriching the available datasets for DDA, the CIR implementation boosts the discoverability, extensibility and manageability of the DDA infrastructure. In particular:

- **Discoverability**: CIR enables the discoverability of all registered objects and helps third party applications to combine the information provided from different systems and databases. To this end, a global unique identifier (in a Universally Unique Identifier (UUID) format) is assigned to each registered object.
- **Extensibility**: CIR facilitates the flexible extension of the digital models' infrastructure with information (data/metadata) stemming for additional repositories and databases. This requires however that each new repository extends the data and metadata of the digital models and links them to other models through the CIR at the time of their deployment.
- **Manageability**: Figure 4 depicts a "Data Management Console" as a core element of the infrastructure. This console is aimed at enabling the configuration of the CIR and the databases of the infrastructure through a single point access.

These attributes are in addition to the DDA engine's configurability and can enable very versatile deployments in different industrial environments.

4. Implementation and Validation

4.1. Open Source Implementation

The DDA engine has been implemented as open source software and it available at GitHub at: https://github.com/far-edge/distributed-data-analytics. This open source version comes with distinct modules for the EA-Engine, the DA-Engine and the Open API for analytics. Moreover, it provides some utilities that simulate the operation of the DR&PR component in order to ease the development of demonstrators based on simulation of data streams. The implementation comes with a dashboard that enables management of the DDA deployment, including management of EA-Engine instances and the processors that they comprise. Through the dashboard, application developers and deployer can configure most of the parameters of the DDA, including the analytics functions of the analytics processors, the parameters of the storage processors and of the preprocessors, the deployment of EA-Engines in various edge gateways and more. Two snapshots of the dashboard are depicted in Figure 7. They correspond two different customizations of the dashboard for different projects

(i.e., H2020 FAR-EDGE, H2020 PROPHSY) and use cases, in different plants that comprise distinct sets of data sources. Note that the FAR-EDGE dashboard is also illustrated in [24].

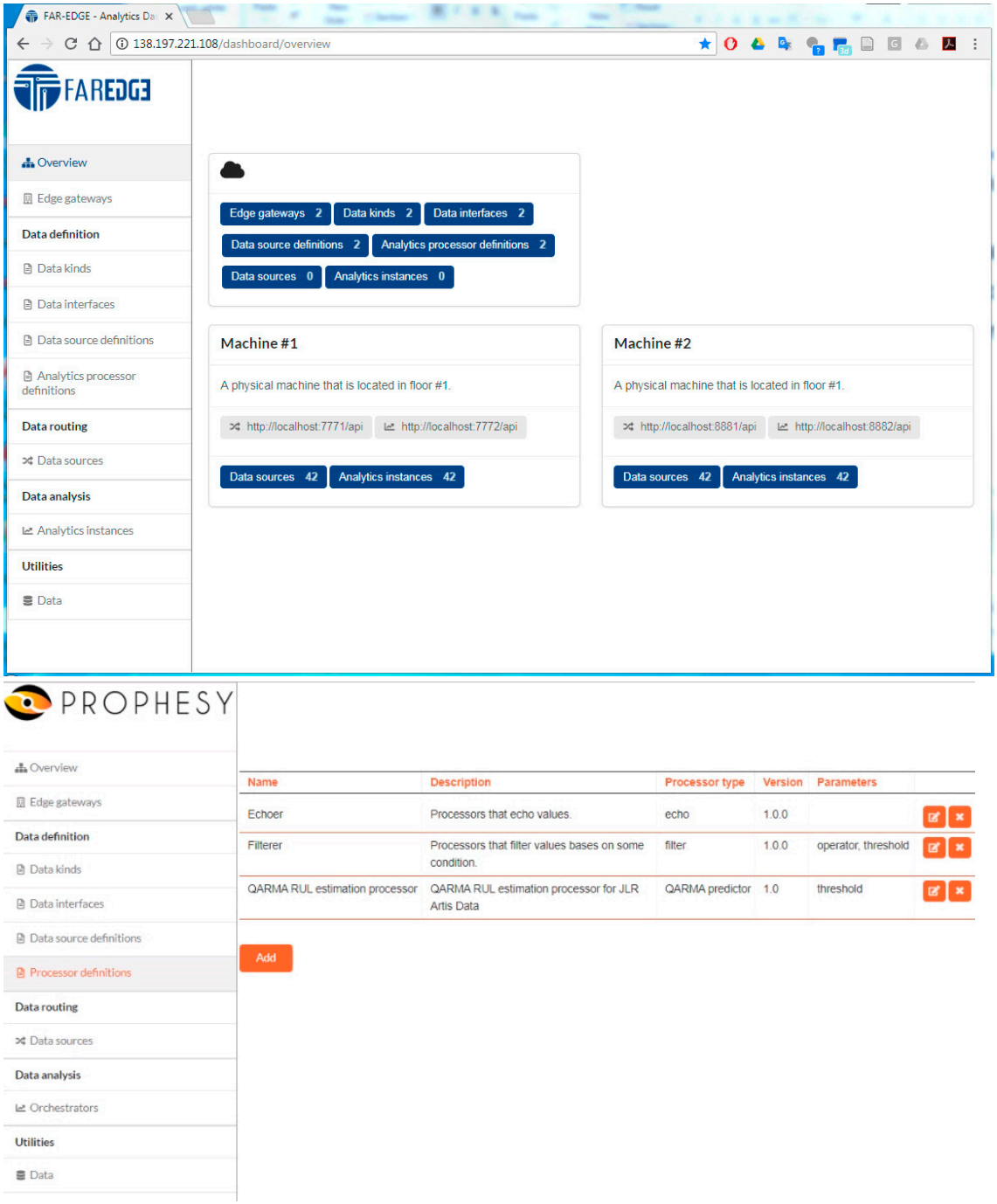


Figure 7. Dashboard for managing the EA-Engine Implementation.

Note that the implementation of the CIR registry is not provided within the repository of the DDA's open source software. It is released as a separate software/middleware module, as it can be used to link databases and objects independently of the DDA.

In order to validate the operation of the DDA and our claims about its configurability, extensibility and versatility, we deploy it for three distinct analytics applications in a different industrial environment. The applications concern some of the most common IIoT applications in Industry 4.0.

4.2. Complete Analytics Modelling Scenarios in a Pilot Production Line: Validation in Power Consumption Analytics

The DDA has been deployed on a pilot line (testbed) in order to analyze and calculate power consumption statistics about a set of distributed Infrastructure Boxes (IB). In particular, the DDA was configured to handle data streams (i.e., processors) calculating the hourly average power and the hourly average energy for: (i) Each IB based on an EA-Engine deployment for each one; and (ii) Across multiple IBs based on a DA-Engine deployment. The calculation workflows included tests of the calculated values against thresholds in order to identify anomalies during production, while at the same time, enabling the pilot line operators to better understand the machine's behavior and the overall responsiveness of each production process.

Two different scenarios have been deployed over different Infrastructure Boxes (IB) (i.e., modules) of the pilot line:

- The first scenario utilizes only one Edge Gateway with the Edge Analytics, and
- The second utilizes the full-fledged deployment and of the DDA infrastructure i.e., operates across multiple edge gateways.

For the Edge Analytics, we provide the hourly daily consumption from an Infrastructure Box for two parameters, namely the total power consumption (TotalRealPower) and the total energy consumption (TotalRealEnergy). Likewise, for the Distributed Analytics, we also provide the hourly daily consumption from all Infrastructure Boxes for the same parameters (TotalRealPower and TotalRealEnergy).

Each of the two scenarios has been modelled and represented in the digital world based on the FAR-EDGE Data Models (i.e., and their corresponding schemas), as illustrated in following paragraphs.

In order to set up the Edge Analytics infrastructure and to model the Edge Analytics functionalities in the digital world, one must undertake the following steps:

- **Step 1 (IB Modelling)**: One Edge Gateway is deployed/bind with each Infrastructure Box. The Infrastructure Box is modelled using the following constructs of the data models infrastructure: (i) The Data Interface (DI) (Table 1); (ii) The Data Source (DSD) (Table 2); (iii) The Data Kind (DK) (Table 3). This information is stored in the Data Model repository of the DDA, which is deployed in the cloud.
- **Step 2 (IB Instantiation/Registration)**: The data models of the IB are used to generate the Data Source Manifest (DSM) (Table 4) and register it to each Edge Gateway.
- **Step 3 (Edge Analytics Modelling)**: The required processors are modelled with the help of an Analytics Processor Definition (APD) object (Table 5). They include: (i) A processor for hourly average calculation from a single data stream; (ii) A processor for persisting results in a nonSQL repository (i.e., a MongoDB in our case). This information is also stored at the data model repository of the DDA.
- **Step 4 (Edge Analytics Installation/Registration)**: Based on the data models specified in the previous steps, it is possible to generate the Analytics Processor Manifest (APM) for each required processor which is registered to the Edge Gateway, in particular: (i) The processor for hourly average calculation from the TotalRealPower data stream; (ii) The processor for hourly average calculation from the TotalRealEnergy data stream; (iii) The processor for persisting results in a database (i.e., MongoDB in our case) of the Edge Gateway, which is to be used by the edge analytics; (iv) The processor for persisting results in the cloud, which is to based used for distributed analytics. Furthermore, a, Analytics Manifest (AM) (Table 6) is generated for orchestrating the instantiated processors. The AM is registered and started through the Edge Gateway Analytics Engine API.

Table 1. Data Interface (DI) Modelling.


```
{
  "_id": "c64709bb-6565-41fe-8256-f6802ad9a538",
  "name": "MQTT topic",
  "communicationProtocol": "MQTT",
  "parameters": {
    "parameter": [
      {
        "name": "host",
        "description": "The host where the MQTT broker runs.",
        "dataType": "string",
        "defaultValue": "localhost"
      },
      {
        "name": "port",
        "description": "The port where the MQTT broker listens.",
        "dataType": "int",
        "defaultValue": 1883
      },
      {
        "name": "username",
        "description": "The username to use to connect to the MQTT broker.",
        "dataType": "string"
      },
      {
        "name": "password",
        "description": "The password to use to connect to the MQTT broker.",
        "dataType": "string"
      },
      {
        "name": "topic",
        "description": "The MQTT topic.",
        "dataType": "string"
      }
    ]
  }
}
```


Table 2. Data Source Definition (DSD) Modelling.


```
{
  "_id": "e810e6cb-f93e-4c1d-8365-2dda74b63bb8",
  "name": "Total real energy in JSON format over MQTT",
  "dataInterfaceReferenceID": "c64709bb-6565-41fe-8256-f6802ad9a538",
  "dataKindReferenceIDs": {
    "dataKindReferenceID": [
      "5071a52f-be67-4d21-bddb-24de8b6144a7"
    ]
  }
}
```


Table 3. Data Kind (DK) Modelling.


```
{
  "_id": "5071a52f-be67-4d21-bddb-24de8b6144a7",
  "name": "Total real energy in plain text",
  "description": "Total real energy values (in KWh) in plain text.",
  "format": "Plain text",
  "quantityKind": "Energy"
}
```

Table 4. Data Source Manifest (DSM) Modelling.


```
{
   "id": "7efa7286-f689-4f51-980c-8e270a1c6d7d",
   "macAddress": "8c:85:90:96:40:cd",
   "dataSourceDefinitionReferenceID": "e810e6cb-f93e-4c1d-8365-2dda74b63bb8",
   "dataSourceDefinitionInterfaceParameters": {
      "parameter": [
         {
            "key": "host",
            "value": "192.168.253.240"
         },
         {
            "key": "port",
            "value": 1883
         },
         {
            "key": "username",
            "value": "faredge"
         },
         {
            "key": "password",
            "value": "*********"
         },
         {
            "key": "topic",
            "value": "PhoenixIB/TotalRealEnergy"
         }
      ]
   }
}
```


Table 5. Modelling of APD for Calculating an Average.


```
{
   "_id": "d5dbd123-0cd4-42bf-aa4f-6c459b4f3060",
   "name": "Average calculator",
   "description": "Processors that read values and calculate their average.",
   "processorType": "average",
   "version": "1.0.0"
}
```

Table 6. Modelling of AM for Orchestrating Processors.

```
{
   "edgeGatewayReferenceID": "da2282f0-4a13-4f68-a1a3-fe05da03c704",
   "analyticsProcessors": {
      "apm": [
         {
            "id": "950354ff-7ce7-4601-a7ea-11f5175d086c",
            "analyticsProcessorDefinitionReferenceID":
                        "d5dbd123-0cd4-42bf-aa4f-6c459b4f3060",
            "dataSources": {
               "dataSource": [
                  {
                     "dataSourceManifestReferenceID":
                           "2fae2556-d516-464d-8b18-aeb2b473e759"
                  }
               ]
            },
            "dataSink": {
               "dataSourceManifestReferenceID": "f006a60b-c40f-4489-a7f7-c73d10d7d0eb"
            }
         }
      ]
   }
}
```

Likewise, in order to set up the distributed analytics, the following steps are followed, in addition to those listed above:

- **Step 5 (Distributed Analytics Modelling)**: In this step the required processors are modelled with the help of an Analytics Processor Definition (APD) object. In particular, the following processors are modelled: (i) A processor for hourly average calculation for values from a database; (ii) A processor for persisting results in the database. The above models are stored in the data model repository at the cloud.
- **Step 6 (Distributed Analytics Installation/Registration)**: The previously specified data models can be used to generate the Analytics Processor Manifest (APM) for each required Processor which is registered to the cloud. These processors include: (i) A for Processor hourly average calculation for TotalRealPower for all Infrastructure Boxes from the global database (i.e., MongoDB in our implementation); (ii) A processor for hourly average calculation from TotalRealEnergy for all Infrastructure Boxes from the global database (i.e., MongoDB in our implementation); (iii) A processor for persisting results in the global database (i.e., MongoDB in our implementation). Moreover, an Analytics Manifest (AM) is modelled and generated for orchestrating the instantiated processors. The AM is registered and started through the Distributed Data Analytics Engine API.

Based on the above listed steps are setting up a configurable infrastructure without any programming effort, given that most of the steps can be undertaken through the dashboard interface. Note also that the various tables present the models in JSON (JavaScript Object Notation). Table 7 presents an instance of the outcome of the analytics operation, modelled in-line with the Digital Models of the Analytics engine (i.e., according to the so-called LiveDataSet object).

Table 7. Sample LiveDataSet for total real Energy Calculation based on the DDA.


```
{
    "observation": [{
        "id": "84190868-8dbb-41ea-8b90-ea8f4699f43a",
        "name": "PhoenixIB-TotalRealEnergy",
        "dataKindReferenceID": "c4b0ecec-0086-46e8-9812-04bde8be2f08",
        "timestamp": "2018-05-24 14:59:14.924",
        "value": 210
    }],
    "id": "1aed50c9-d507-4e16-b7ca-a44321bc37ac",
    "dataSourceManifestReferenceID": "2fae2556-d516-464d-8b18-aeb2b473e759",
    "mobile": false,
    "timestamp": "2018-05-24 14:59:14.924"
}{
    "observation": [{
        "id": "1dd0666c-b61e-494d-91bc-71ab4a4717c3",
        "name": " PhoenixIB-TotalRealEnergy",
        "dataKindReferenceID": "c4b0ecec-0086-46e8-9812-04bde8be2f08",
        "timestamp": "2018-05-24 15:00:14.931",
        "value": 215
    }],
    "id": "ba98ea74-646d-4519-bb9e-57a439acb82b",
    "dataSourceManifestReferenceID": "2fae2556-d516-464d-8b18-aeb2b473e759",
    "mobile": false,
    "timestamp": "2018-05-24 15:00:14.931"
}{
    "observation": [{
        "id": "14358864-c768-4bc2-bdb3-19aa88d482e1",
        "name": " PhoenixIB-TotalRealEnergy",
        "dataKindReferenceID": "c4b0ecec-0086-46e8-9812-04bde8be2f08",
        "timestamp": "2018-05-24 15:01:14.95",
        "value": 205
    }],
    "id": "f2454dad-a6b9-43c1-a741-fbeeebf96b5b",
    "dataSourceManifestReferenceID": "2fae2556-d516-464d-8b18-aeb2b473e759",
    "mobile": false,
    "timestamp": "2018-05-24 15:01:14.95"
}
```


In following paragraphs, we also discuss the deployment and use of the DDA in other use cases in real plants. However, the discussion of these use cases is not accompanied by the JSON models of the corresponding analytics processes, given that the deployments are proprietary and much more complex that the one of the pilot plant which renders the detailed description of the data sources and of the analytics manifests beyond the scope of this manuscript.

4.3. Validation in Real Production Lines

4.3.1. Validation in Mass Customization

Mass customization is one of the most prominent Industry 4.0 applications, which is based on the use of IIoT for building flexible production lines that produce diversified products. Estimating the time of production operations for very different products is essential for ensuring the effectiveness of mass customization operations. Factual calculations based on statistical models that exploit digital data from past production operations are therefore preferred over empirical estimations, as they improve the accuracy of the production process and boost the organization and the efficiency of the mass customization process. In this context, the DDA has been configured and deployed in order to calculate assembly times of trucks in a mass customization production line. The DDA has been configured

to operate at the level of a station, in order to provide fast, automated and accurate calculations of assembly times based on past data. Such calculations are also beneficial for the operators of the station, as they enhance their knowledge about the complexity and timing of the assembly process.

The DDA has been provided with parameters of past assembly operations and their timing, including a diverse set of parameters that are a direct results of the product variation (e.g., number of axles of the truck, radars and cameras to be calibrated for a specific track, number and types of extra protection plates to be mounted). EA-Engine instances have been configured and deployed to calculate: (i) Planned process time for different truck types based on historic data about the assembly times of similar variants; (ii) In Line Adjustment Probability per truck type, based on statistical processing over known cases of adjustment and (iii) Estimated Adjustment Times for different types and variations of the product based on previously known estimates about produced trucks.

4.3.2. Validation in Predictive Maintenance

A second deployment of the DDA has taken place in a predictive maintenance scenario. As part of this deployment, a diverse number of data streams are configured and processed through the DDA, including dynamic streaming data about the production line and the process (e.g., temperature, acoustic data from vibration), information about failure modes and maintenance actions associated with the machine and the process, as well as data associated with the quality of the product. The deployment involves processing of data from a variety of different systems, sensors and databases, which were modelled as data sources. Using the project's DSL, an analytics pipeline was configured that comprised preprocessors preparing the data for consumption by Machine Learning (ML) algorithms, as well as analytics processors implementing these algorithms. The major outcome of the DDA deployment concerns the calculation of the RUL (Remaining Useful Life) of the equipment.

Overall, the deployment of the DDA infrastructure in diverse industrial settings and use cases is indicative of its configurability and versatility, beyond a baseline technical validation of the DDA's operation.

5. Conclusions and Future Work

Most IIoT use cases are that are data intensive and require tools and techniques for efficient Distributed Data Analytics. The latter are already among the main pillars of Industry 4.0 deployments in applications like flexible automation, predictive maintenance and digital simulations [26]. Effective IIoT analytics are in most cases associated with the handling of heterogeneous streaming data i.e., data streams with very high ingestion rates. State of the art streaming analytics engines provide exceptional performance and low latency, as they can be deployed in modern clusters in ways that exploit parallelization of tasks. Less emphasis, however, has been paid in their configurability and their integration in scenarios where multiple streaming systems must be combined in a single deployment.

In this paper, we have introduced a configurable streaming engine, as a meta-level engine that can combine data streams from diverse streaming middleware platforms. The presented engine can be flexibly configured to operate in modern cloud/edge environments. It offers a DSL that facilitates the configuration of data sources, along with their integration, preprocessing and analysis based on a variety of user defined analytical functions. The engine leverages the performance of state-of-the-art streaming platforms like Kafka, while it can be flexibly configured in different IIoT environments. The latter configurability has been validated in different deployment scenarios in applications like predictive maintenance, production planning for mass customization and detection of anomalies in the operation of machines. It is also important that the implementation of IIoT DDA infrastructure is provided as open source software and is available for deployment, use and further extension by the IIoT developers' community.

The introduced DDA engine has already been used in three distinct deployments in pilot production lines (i.e., testbeds), as well as in real life production lines (i.e., realistic manufacturing environments). Furthermore, we will plan to use it for data analytics applications in two more factories

dealing with automotive and glass products. These deployments have already proven the generality of the platform and its potential to accelerate IIoT data streaming deployments in real-life settings. Moreover, they have provided us with the opportunity to improve the implementation in terms of robustness and performance. Our future plans include the implementation of tools for managing the data models and the DSL of the DDA in a visual and integrated way. We plan to evolve the dashboard implementation to a more complete programming environment, which will be flexibly integrated with other tools, such as open sources tools for data mining and data analytics. Likewise, based on the experience of the practical deployments, we plan to create an initial pool of commonly used analytics processors that could accelerate the development of analytics applications through reuse. In these ways, the DDA will gradually evolve from a research prototype to a more robust platform for industrial use.

Author Contributions: Conceptualization, N.K., A.R. and J.S.; Methodology, N.K.; Software, N.K.; Validation, A.R.; Resources, A.R.; Data curation, N.K.; Writing—original draft preparation, J.S.; Supervision, J.S.

Funding: This research was funded by the European Commission, grant number 720395 and grant number 766994. Both grants are funded as part of the European Commission's H2020 programme.

Acknowledgments: Part of this work (namely the distributed analytics engine design and implementation) has been carried out in the scope of the H2020 FAR-EDGE project (contract number 723094), while another part (namely the common interoperability registry design and implementation) has been carried out in the scope of H2020 PROPHESY project (contract number 766994). Both projects are funded by the European Commission in the scope of its H2020 programme. The authors acknowledge valuable help and contributions from all partners of both projects, including the plants where the DDA infrastructure has been validated.

The present manuscript is a significantly extended and enhanced version of the three pages poster paper that has been presented in the 2019 edition of IEEE International Conference on Distributed Computing in Sensor Systems [26].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Abadi, D.; Carney, D.; Çetintemel, U.; Cherniack, M.; Convey, C.; Lee, S.; Stonebraker, M.; Tatbul, N.; Zdonik, S. Aurora: A new model and architecture for data stream management. *VLDB J.* **2003**, *12*, 120–139. [CrossRef]
2. Chandrasekaran, S.; Cooper, O.; Deshpande, A.; Franklin, M.; Hellerstein, J.; Hong, W.; Krishnamurthy, S.; Madden; Reiss, F.; Shah, M. TelegraphCQ: Continuous dataflow processing. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03), New York, NY, USA, 9–12 June 2003; p. 668.
3. Arasu, A.; Babu, S.; Widom, J. The CQL continuous query language: Semantic foundations and query execution. *VLDB J.* **2006**, *15*, 121–142. [CrossRef]
4. Ahmad, Y.; Berg, B.; Cetintemrel, U.; Humphrey, M.; Hwang, J.; Jhingran, A.; Maskey, A.; Papaemmanouil, O.; Rasin, A.; Tatbul, N.; et al. Distributed operation in the Borealis stream processing engine. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '05), New York, NY, USA, 14–16 June 2005; pp. 882–884.
5. Biem, A.; Bouillet, E.; Feng, H.; Ranganathan, A.; Riabov, A.; Verscheure, O.; Koutsopoulos, H.; Moran, C. IBM infosphere streams for scalable, real-time, intelligent transportation services. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '10), New York, NY, USA, 6–10 June 2010; pp. 1093–1104.
6. Gulisano, V.; Jiménez-Peris, R.; Patiño-Martínez, M.; Soriente, C.; Valduriez, P. StreamCloud: An Elastic and Scalable Data Streaming System. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 2351–2365. [CrossRef]
7. Zaharia, M.; Xin, R.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.; et al. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
8. Carbone, P.; Katsifodimos, A.; Ewen, S.; Markl, V.; Haridi, S.; Tzoumas, K.K. Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull.* **2015**, *38*, 28–38.
9. Noghabi, S.; Paramasivam, K.; Pan, Y.; Ramesh, N.; Bringhurst, J.; Gupta, I.; Campbell, R. Samza: Stateful scalable stream processing at Linked. *Proc. VLDB Endow.* **2017**, *10*, 1634–1645. [CrossRef]

10. Murray, D.; McSherry, F.; Isaacs, R.; Isard, M.; Barham, P.; Abadi, M. Naiad: A timely dataflow system. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13), New York, NY, USA, 3–6 November 2013; pp. 439–455.
11. Kreps, J.; Narkhede, N.; Rao, J. Kafka: A Distributed Messaging System for Log Processing. In Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece, 12–16 June 2011.
12. Isaja, M.; Soldatos, J.; Gezer, V. Combining Edge Computing and Blockchains for Flexibility and Performance in Industrial Automation. In Proceedings of the Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), Barcelona, Spain, 12–16 November 2017; pp. 159–164.
13. Kosar, T.; Bohrab, S.; Mernika, M. Domain-Specific Languages: A Systematic Mapping Study. *Inf. Softw. Technol.* **2016**, *71*, 77–91. [CrossRef]
14. Mernik, M.; Heering, J.; Sloane, A. When and how to develop domain-specific languages. *ACM Comput. Surv.* **2005**, *37*, 316–344. [CrossRef]
15. Anagnostopoulos, A.; Soldatos, J.; Michalakos, S. REFiLL: A lightweight programmable middleware platform for cost effective RFID application development. *Pervasive Mob. Comput.* **2009**, *5*, 49–63. [CrossRef]
16. Kefalakis, N.; Soldatos, J.; Konstantinou, N.; Prasad, N. APDL: A reference XML schema for process-centered definition of RFID solutions. *J. Syst. Softw.* **2011**, *84*, 1244–1259. [CrossRef]
17. Kosar, T.; López, P.M.; Barrientos, P.; Mernik, M. A preliminary study on various implementation approaches of domain-specific language. *Inf. Softw. Technol.* **2008**, *50*, 390–405. [CrossRef]
18. Johanson, A.N.; Hasselbring, W. Effectiveness and efficiency of a domain-specific language for high-performance marine ecosystem simulation: A controlled experiment. *Empir. Softw. Eng.* **2017**, *22*, 2206–2236. [CrossRef]
19. Kosar, T.; Gaberc, S.; Carver, J.; Mernik, M. Program comprehension of domain-specific and general-purpose languages: Replication of a family of experiments using integrated development environments. *Empir. Softw. Eng.* **2018**, *23*, 2734–2763. [CrossRef]
20. Silva, A. Model-driven engineering: A survey supported by a unified conceptual model. *Comput. Lang. Syst. Struct.* **2015**, *43*, 139–155.
21. Lelandais, B.; Oudot, M.; Combemale, B. Applying Model-Driven Engineering to High-Performance Computing: Experience Report, Lessons Learned, and Remaining Challenges. *J. Comput. Lang.* **2019**, *55*, 1–19. [CrossRef]
22. Petrali, P.; Isaja, M.; Soldatos, J. *Edge Computing and Distributed Ledger Technologies for Flexible Production Lines: A White-Appliances Industry Case*; In IFAC: Geneva, Switzerland, 2018; Volume 51, pp. 388–392.
23. Soldatos, J.; Lazaro, O.; Cavadini, F. (Eds.) The Digital Shopfloor: Industrial Automation in the Industry 4.0 Era Forthcoming Performance Analysis and Applications. River Publishers Series in Automation, Control and Robotics, ISBN 9788770220415, e-ISBN 9788770220408. February 2019. Available online: https://www.riverpublishers.com/book_details.php?book_id=676 (accessed on 18 November 2019).
24. Kefalakis, N.; Roukounaki, A.; Soldatos, J. A Configurable Distributed Data Analytics Infrastructure for the Industrial Internet of things. In Proceedings of the DCOSS, Santorini Island, Greece, 29–31 May 2019; pp. 179–181.
25. Mathew, A.; Zhang, L.; Zhang, S.; Ma, L. A Review of the MIMOSA OSA-EAI Database for Condition Monitoring Systems. In *Engineering Asset Management*; Mathew, J., Kennedy, J., Ma, L., Tan, A., Anderson, D., Eds.; Springer: London, UK, 2006.
26. Soldatos, J. Introduction to Industry 4.0 and the Digital Shopfloor Vision. In *The Digital Shopfloor: Industrial Automation in the Industry 4.0 Era*; River Publishers: Delft, The Netherlands, 2019; ISBN 108770220417.