Edge Computing and Blockchains for Flexible and Programmable Analytics in Industrial Automation

Mauro Isaja
Research & Development
Engineering Ingegneria Informatica SpA
Rome, Italy
e-mail: mauro.isaja@eng.it

John Soldatos
IoT Group
Athens Information Technology
Maroussi, Greece
e-mail: jsol@ait.gr

Nikos Kefalakis
IoT Group
Athens Information Technology
Maroussi, Greece
e-mail: nkef@ait.gr

Volkan Gezer
Innovative Factory Systems (IFS)
DFKI
Kaiserslautern, Germany
e-mail: Volkan.Gezer@dfki.de

Abstract **- The advent of Industry 4.0 has given rise to the introduction of new industrial automation architectures that emphasize the use of digital technologies. In this paper, we present a novel, standards-based Reference Architecture for industrial automation, which combines the benefits of edge computing and blockchain technologies for flexible and reliable orchestration of automation workflows and distributed data analytics. Accordingly, we illustrate a practical implementation of the Reference Architecture for scalable and programmable data analytics, along with its deployment in an Industry 4.0 pilot plant.**

Keywords-component; Factory automation; edge computing; blockchain; RAMI4.0; IIRA; Industry4.0; distributed data analytics; programmability; distributed ledger

I. INTRODUCTION

The vision of future manufacturing foresees flexible and hyper-efficient plants that will enable manufacturers to support the transition from conventional "made-to-stock" production models, to the emerging customized ones such as "made-to-order", "configure-to-order" and "engineering-to-order" [1]. Flexibility in automation is a key prerequisite to supporting the latter production models: It facilitates manufacturers to change automation configurations and rapidly adopt new automation technologies, as a means of supporting variation in production without any essential increase in production costs.

In order to support flexibility in automation, the industrial automation community has been exploring options for the virtualization of the automation pyramid, as part of the transformation of mainstream centralized automation models (like ISA-95) to more distributed ones. Several research and development initiatives have introduced decentralized factory automation solutions based on technologies like intelligent agents [2][3] and Service Oriented Architectures (SOA) [4][5]. These initiatives produced proof-of-concept implementations that highlighted the benefits of decentralized automation in terms of flexibility. However, they are still not being widely deployed in manufacturing plants, mainly due to the fact that the cost-benefit ratio of such solutions is perceived as unfavourable. Nevertheless, the vision of decentralizing the factory automation pyramid is still alive, as this virtualization can potentially make production systems more agile, increase product quality and reduce cost.

With the advent of the fourth industrial revolution (Industry 4.0) and the Industrial Internet of Things (IIoT), decentralization is being revisited in the light of the integration of Cyber-Physical Systems (CPS) with cloud computing infrastructures. Therefore, several cloud-based applications are deployed and used in factories, which leverage the capacity and scalability of the cloud while fostering supply chain collaboration and virtual manufacturing chains. Early implementations have also revealed the limitations of the cloud in terms of efficient bandwidth usage and its ability to support real-time operations, including operations close to the field.

More recently, the edge computing paradigm has been explored in order to alleviate the limitations of cloud-centric architectures. Edge computing architectures move some part of the system's overall computing power from the cloud to its edge nodes, i.e., on the field or in close proximity to it –as a means of [6][7]:

- Saving bandwidth and storage, as edge nodes can filter data streams from the field in order to get rid of information without value for industrial automation.
- Enabling low-latency and proximity processing, since information can be processed close to the field.
- Providing enhanced scalability, through supporting decentralized storage and processing that scales better than cloud processing.
- Supporting shopfloor isolation and privacy-friendliness, since edge nodes at the shopfloor are isolated from the rest of the network.

These benefits make edge computing suitable for specific classes of use cases in factories, including:

- Large scale distributed applications, typically applications that involve multiple plants or factories, which process streams from numerous devices at scale.
- Near-real-time applications, which analyse data close to the field or even control CPS systems such as smart machines and industrial robots.

As a result, the application of edge computing to factory automation is extremely promising, since it empowers decentralization in a way that still supports real-time interactions and scalable analytics. Therefore, it is no accident that there are ongoing efforts to provide edge computing implementations for industrial automation in general and factory automation in particular. Furthermore, reference architectures (RAs) for IIoT and industrial automation exist, which highlight the importance of edge computing for compliant implementations. In this article, we present a reference architecture (RA) for factory automation based on edge computing and distributed ledger technology, which has been specified as part of the H2020 FAR-EDGE project [8]. The FAR-EDGE RA specifies some unique features and capabilities, which differentiate them from other on-going implementations of edge computing for factory automation. Most of these unique features concern the exploitation of Distributed Ledger Technology (DLT), today commonly referred to as "blockchain", as a means of representing and synchronizing automation and data analytics processes based on Smart Contracts. These can be dynamically configured, stored securely and executed in a distributed way, enabling flexibility and scalability in factory automation processes. The implementation of such smart contracts can take advantage of existing distributed ledger platforms. In the scope of FAR-EDGE the popular, open source Hyperledger Fabric provides the foundation for implementing smart contracts and synchronizing distributed processes, as presented in later sections.

Overall, the FAR-EDGE RA combines the power of edge computing for performing operations close to the field, with the reliability and trustworthiness benefits of distributed ledger technologies in terms of the synchronization of distributed processes. This is based on the implementation of a tier of edge nodes where edge functionalities are performed, and its combination with a tier of ledger services that is in charge of plant-wide synchronization of edge nodes. As part of the paper we illustrate the implementation of a Distributed Data Analytics (DDA) platform that adheres to the FAR-EDGE RA, leveraging both edge and ledger tier functionalities. This DDA implementation comes with an additional benefit: it is flexibly extensible and programmable in terms of the definition of edge processing capabilities over field data. Combined with the functionalities of the ledger services, this programmability provides integrators of industrial automation solutions with additional flexibility in implementing distributed analytics systems. The implementations of both the ledger services and the programmable Edge Analytics (EA) services are publicly available as open source software.

Note that the present paper represents a significantly extended version of a conference paper that introduced the FAR-EDGE RA [1]. In particular, this paper includes practical insights on the actual implementation of the edge and ledger tiers of the RA, as part of the DDA platform. It is therefore targeted to researchers and practitioners that might be interested in using the RA and/or its open source implementation in their solutions. It is also destined to the IIoT and Industry4.0 open source community, which is starving for novel, yet practical components for industrial automation and distributed analytics. Note that this paper is a significantly extended version of a conference article of the authors [1].

The paper is structured as follows: Section II, following this introduction, presents state-of-the-art specifications and implementations of the edge computing paradigm for factory automation, as well as the current status in the use of blockchains for industrial applications. It also positions FAR-EDGE against them. Section III introduces the FAR-EDGE RA, from a functional and structural perspective. Section IV presents the prototype implementation of the DDA platform, including its programmability features. Section V is devoted to the presentation of the implementation of ledger services, based on extensions over a permissioned blockchain infrastructure, namely IBM's Hyperledger Fabric. Section VI illustrates the deployment of the EA systems in an Industry4.0 pilot plant. Finally, Section VII concludes the paper.

II. Related Work

Acknowledging the benefits of edge computing for industrial automation, standards development organizations (SDOs) have specified relevant RAs, while industrial organizations are already working towards providing tangible edge computing implementations.

SDOs such as the OpenFog Consortium and the Industrial Internet Consortium (IIC) have produced RAs for industrial automation applications. In particular, the RA of the OpenFog Consortium prescribes a high-level architecture for Internet-of-Things (IoT) systems, which covers industrial IoT use cases. On the other hand, the RA of the IIC [9] outlines the structuring principles of systems for industrial applications. The IIC RA prescribes the use of edge computing components and principles for compliant implementations. It addresses a wide range of industrial use cases in multiple sectors, including factory automation. These RAs have been recently released and their reference implementations are still in their early stages.

A reference implementation of the IIC RA's edge computing functionalities for factory automation is provided as part of IIC's edge intelligence testbed [10]. This testbed provides a proof-of-concept implementation of edge computing functionalities on the shopfloor. The focus of the testbed is on configurable edge computing environments, which enable the development and testing of systems and algorithms for EA. Moreover, Dell-EMC has recently announced the EdgeX Foundry framework [11], which is a vendor-neutral open source project hosted by the Linux Foundation that builds a common open framework for IIoT

edge computing. The framework is influenced by the above-listed RAs and was recently released. Other vendors are also incorporating support for edge devices and Edge Gateways in their cloud platforms.

FAR-EDGE is uniquely positioned in the landscape of edge computing solutions for factory automation. In particular, the FAR-EDGE architecture is aligned to the IIC RA, while exploiting concepts from other RAs and standards such as the OpenFog RA and RAMI 4.0 (Reference Architecture Model Industry 4.0) [12]. However, FAR-EDGE explores pathways and offers functionalities that are not addressed by other specifications and reference implementations. In particular, it researches the applicability of disruptive key enabling technologies like DLT and Smart Contracts in factory automation. DLT, while being well understood and thoroughly tested in mission-critical areas like digital currencies (e.g., Bitcoin), have never been applied before to industrial systems. This is mainly due to performance concerns about their use, despite their trustworthiness and reliability benefits. However, in literature the merit of DLT for synchronizing distributed processes have been recently acknowledged [13].

FAR-EDGE aims at demonstrating how a pool of specific Ledger Services built on a generic DLT platform can enable decentralized factory automation in an effective, reliable, scalable and secure way. In particular, Ledger Services are responsible for sharing process state and enforcing business rules across the computing nodes of a distributed system, thus permitting virtual automation and analytics processes that span multiple nodes – or, from a bottom-up perspective, autonomous nodes that cooperate to a common goal. This is one of project's unique contributions, which sets it apart from similar edge computing efforts and provides increased flexibility and reliability.

III. FAR-EDGE REFERENCE ARCHITECTURE

The FAR-EDGE RA is a conceptual framework that drives the design and the implementation of automation platforms based on edge computing and DLT technologies. It is aligned to IIC's RA concepts and described from two architectural viewpoints: the functional viewpoint and the structural viewpoint, as outlined in following paragraphs. An overall architecture representation that includes all elements is provided in Figure 1.

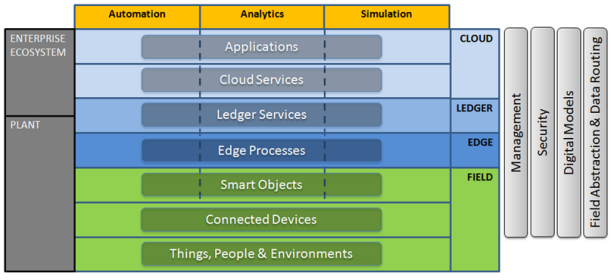

Figure 1. Overview of the FAR-EDGE RA

A. *Functional Viewpoint*

According to the FAR-EDGE RA, the functionality of a factory automation platform can be decomposed into three high-level Functional Domains - Automation, Analytics and Simulation – and four Crosscutting (XC) Functions – Management, Security, Digital Models and Field Abstraction & Data Routing. To better clarify the scope of such topics, we have tried to map them to similar Industrial Internet RA (IIRA) concepts [9]. Functional Domains and XC Functions are orthogonal to structural Tiers: the implementation of a given functionality may – but is not required to – span multiple Tiers, so that in the overall architecture representation Functional Domains appear as vertical lanes drawn across horizontal layers. Figure 2 highlights the relationship between Functional Domains, their users and the factory environment. It also uses arrows to show the flow of data and of control.

Automation Domain: The FAR-EDGE Automation domain includes functionalities supporting automated control and automated configuration of physical production processes. Automated configuration is the enabler of plug-and-play factory equipment (better known as plug-and-produce), which in turn is a key technology for mass-customization, as it allows a faster and less expensive adjustments of the production process. The Automation domain requires a bidirectional monitoring/control communication channel with the Field, typically with low bandwidth but very strict timing requirements. In some advanced scenarios, Automation is controlled – to some extent – by the results of Analytics and/or Simulation. The Automation domain partially maps to the Control domain of the IIRA.

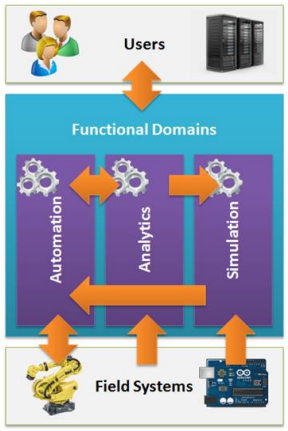

Figure 2. FAR-EDGE RA Functional Domains

Analytics Domain: The FAR-EDGE Analytics domain includes functionalities for gathering and processing Field data for a better understanding of production processes, i.e., a factory-focused business intelligence. This typically requires a high-bandwidth Field communication channel, as the volume of information that needs to be transferred in a given time unit may be substantial. On the other hand, channel latency tends to be less critical than in the Automation scenario. The Analytics domain provides

intelligence to its users, but these are not necessarily limited to humans or vertical applications (e.g., a predictive maintenance solution). In particular, the Automation and Simulation domains, if properly configured, can both make direct use of the outcome of data analysis algorithms. In the case of Automation, the behaviour of a workflow might change in response to changes detected in the controlled process – e.g., a process drift caused by the progressive wear of machinery or by the quality of assembly components being lower than usual. In the case of Simulation, data analysis can be used to update the parameters of a digital model. The Analytics domain matches perfectly the Information domain of the IIRA, except that the latter is receiving data from the Field through the mediation of Control functionalities.

Simulation Domain: The FAR-EDGE Simulation domain includes functionalities for simulating the behaviour of physical production processes for the purpose of optimization or of testing what/if scenarios at minimal cost and risk and without any impact of regular shop activities. Simulation requires digital models of plants and processes to be in-sync with the real-world objects they represent. As the real world is subject to change, models should reflect those changes. For instance, the model of a machine assumes a given value of electric power / energy consumption, but the actual values will diverge as the real machine wears down. To detect this gap and correct the model accordingly, raw data from the Field (direct) or complex analysis algorithms (from Analytics) can be used.

Crosscutting Functions: Crosscutting Functions address common specific concerns. Their implementation affects several Functional Domains and Tiers. They include.

- **Management**: Low-level functions for monitoring and commissioning/decommissioning of individual system modules.
- **Security**: Functions securing the system against the unruly behaviour of its user and of connected systems. These include digital identity management and authentication, access control policy management and enforcement, communication and data encryption.
- **Digital Models**: Functions for the management of digital models and their synchronization with the real-world entities they represent. Digital modes are a shared asset, as they may be used as the basis for automated configuration, simulation and field abstraction – e.g., semantic interoperability of heterogeneous field systems.
- **Field Abstraction & Data Routing**: Functions that ensure the connectivity of business logic (FAR-EDGE RA Functional Domains) to the Field, abstracting away the technical details – like device discovery and communication protocols. Data routing refers to the capability of establishing direct producer-consumer channels on demand, optimized for unidirectional massive data streaming – e.g., for feeding Analytics.

B. Structural Viewpoint

The FAR-EDGE RA uses two classes of concepts for describing the structure of a system: Scopes and Tiers.

Scopes are very simple and straightforward: they define a coarse mapping of system elements to either the factory - Plant Scope - or the broader world of corporate IT - Enterprise Ecosystem Scope. Examples of elements in Plant Scope are machinery, Field devices, workstations, SCADA and MES systems, and any software running in the factory data centre. The Enterprise Ecosystem Scope comprises ERP (Enterprise Resource Planning) and PLM (Product Lifecycle Management) systems and any application or service shared across multiple factories or even companies – e.g., supply chain members.

Tiers are a more detailed and technical-oriented classification of deployment concerns. They can be easily mapped to scopes, but they provide more insight into the relationship between system components. This kind of classification is quite similar to OpenFog RA deployment viewpoint, except for the fact that FAR-EDGE Tiers are industry-oriented while OpenFog ones are not. FAR-EDGE Tiers are one of the most innovative traits of the project's RA, and are described in following paragraphs.

The Field Tier is the bottom layer of the FAR-EDGE RA and is populated by Edge Nodes (EN), i.e., any kind of device that is connected to the digital world on one side and to the real world to the other. ENs can have embedded intelligence (e.g., a smart machine) or not (e.g., a sensor or actuator). The FAR-EDGE RA honours this difference: Smart Objects are ENs with on board computing capabilities, Connected Devices are those without. The Smart Object is where local control logic runs: it is a semi-autonomous entity that does not need to interact frequently with the upper layers of the system. As shown in Figure 3. ENs is actually located over field devices.

The Field is also populated by entities of the real world, i.e., those physical elements of production processes that are not directly connected to the network, and as such are not considered as ENs: Things, People and Environments. These are represented in the digital world by some kind of EN wrapper. For instance, room temperature (Environment) is measured by an IoT sensor (Connected Device), the proximity of a worker (People) to a physical checkpoint location is published by an RFID wearable and detected by an RFID Gate (Connected Device), while a conveyor belt (Thing) is operated by a PLC (Smart Object).

The Field Tier is in Plant Scope. Individual ENs are connected to the digital world in the upper Tiers either directly by means of the shopfloor's LAN, or indirectly through some special-purpose local network (e.g., WSN (Wireless Sensor Network)) that is bridged to the former. From the RAMI 4.0 perspective, the FAR-EDGE Field Tier corresponds to the Field Device and Control Device levels on the Hierarchy axis (IEC-62264/IEC-61512), while the entities there contained are positioned across the Asset and Integration Layers.

The Edge Tier is the core of the FAR-EDGE RA. It hosts those parts of Functional Domains and XC Functions

that can leverage the edge computing model, i.e., software designed to run on multiple, distributed computing nodes placed close to the field, which may include resource constrained nodes. The Edge Tier is populated by Edge Gateways (EG): computing devices that act as a digital world gateway to the real world of the Field. These machines are typically more powerful than the average intelligent EN (e.g., blade servers) and are connected to a fast LAN (Local Area Network). Strategically positioned close to physical systems, the EG can execute Edge Processes: time- and bandwidth-critical functionality having local scope. For instance, the orchestration of a complex physical process that is monitored and operated by a number of sensors, actuators (Connected Devices) and embedded controllers (Smart Objects); or the real-time analysis of a huge volume of live data that is streamed from a nearby Field source.

Deploying computing power and data storage in close proximity to where it is actually used is a standard best practice in the industry. However, this technique basically requires that the scope of individual subsystems is narrow (e.g., a single work station). If instead the critical functionality applies to a wider scenario (e.g., an entire plant or enterprise), it must be either deployed at a higher level (e.g., the Cloud) – thus losing all benefits of proximity – or run as multiple parallel instances, each focused on its own narrow scope. In the latter case, new problems may arise: keeping global variables in-sync across all local instances of a given process, reaching a consensus among local instances on a global truth, collecting aggregated results from independent copies of a data analytics algorithm, etc. The need for peer nodes of a distributed system to mutually exchange information is recognized by the OpenFog RA. A key innovation of the FAR-EDGE approach is that it defines a specific system layer – the Ledger Tier – that is responsible for the implementation of such mechanisms and guarantees an appropriate Quality of Service level.

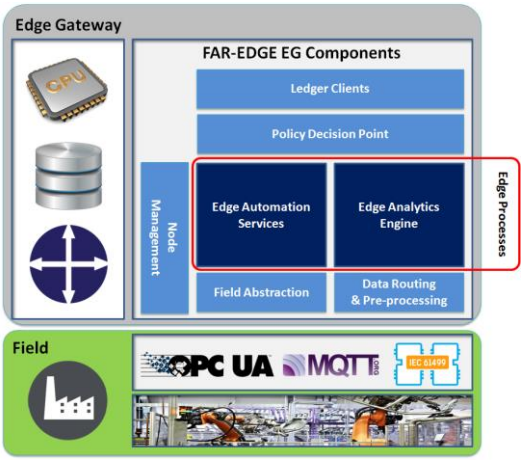

Figure 3. Edge Tier in the FAR-EDGE RA

The Edge Tier is in Plant Scope, located above the Field Tier and below the Cloud Tier. Individual EGs are connected with each other and with the north side of the system, i.e., the globally-scoped digital world in the Cloud Tier – by means of the factory LAN, and to the south side through the shopfloor LAN. From the RAMI 4.0 perspective, the FAR-EDGE Edge Tier corresponds to the Station and Work Centre levels on the Hierarchy axis (IEC-62264/IEC-61512), while the EGs there contained are positioned across the Asset, Integration and Communication Layers. Edge Processes running on EGs, however, map to the Information and Functional Layers.

The Ledger Tier is a complete abstraction: it does not correspond to any physical deployment environment, and even the entities that it "contains" are abstract. Such entities are Ledger Services, which implement decentralized business logic as smart contracts on top of a distributed ledger. Ledger Services are transaction-oriented: each service call that needs to modify the shared state of a system must be evaluated and approved by Peer Nodes before taking effect. Similarly to "regular" services, Ledger Services are implemented as executable code; however, they are not actually executed on any specific computing node: each service call is executed in parallel by all Peer Nodes that happen to be online at the moment, which then need to reach a consensus on its validity. Most importantly, even the executable code of Ledger Services can be deployed and updated online by means of a distributed ledger transaction.

Ledger Services implement the part of Functional Domains and/or XC Functions that enable the edge computing model, through providing support for their Edge Service counterpart. For example, the Analytics Functional Domain may define a local analytics function (Edge Service) that must be executed in parallel on several EGs, and also a corresponding service call (Ledger Service) that will be invoked from the former each time new or updated local results become available, so that all results can converge into an aggregated data set. In this case, aggregation logic is included in the Ledger Service. Another use case may come from the Automation Functional Domain, demonstrating how the Ledger Tier can also be leveraged from the Field: a smart machine with embedded plug-and-produce functionality can ask permission to join the system by making a service call and then, having received green light, can dynamically deploy its own specific Ledger Service for publishing its state and external high-level commands.

The Ledger Tier lays across the Plant and the Enterprise Ecosystem Scopes, as it can provide support to any Tier. The physical location of Peer Nodes, which implement smart contracts and the distributed ledger, is not defined by the FAR-EDGE RA as it depends on implementation choices.

From the RAMI 4.0 perspective, the FAR-EDGE Ledger Tier corresponds to the Work Centre, Enterprise and Connected World levels on the Hierarchy axis (IEC-62264/IEC-61512), while the Ledger Services are positioned across the Information and Functional Layers.

The Cloud Tier is the top layer of the FAR-EDGE RA, and also the simplest and more "traditional" one. It is populated by Cloud Servers (CS): powerful computing machines, sometimes configured as clusters, which are connected to a fast LAN internally to their hosting data centre, and made accessible from the outside world by means of a corporate LAN or the Internet. On CSs runs that part of

the business logic of Functional Domains and XC Functions that benefits from having the widest of scopes over production processes, and can deal with the downside of being physically deployed far away from them. This includes the planning, monitoring and management of entire factories, enterprises and supply chains (e.g., ERP and SCM (Supply Chain Management) systems). The Cloud Tier is populated by Cloud Services and Applications. Cloud Services implement specialized functions that are provided as individual API calls to Applications, which instead "package" a wider set of related operations that are relevant to some higher-level goal and often expose an interactive human interface.

The Cloud Tier is in Enterprise Ecosystem scope. The "Cloud" term in this context implies that Cloud Services and Applications are visible from all Tiers, wherever located. It does not imply that CSs should be actually hosted on some commercial cloud. In large enterprises, the Cloud Tier corresponds to one or more corporate data centres (private cloud), ensuring that the entire system is fully under the control of its owner.

In terms of RAMI 4.0, the FAR-EDGE Cloud Tier corresponds to the Work Centre, Enterprise and Connected World levels on the Hierarchy axis (IEC-62264/IEC-61512), while the Cloud Services and Applications are positioned across the Information, Functional and Business Layers.

IV. Edge Tier Services For High-Performance And Programmable Edge Analytics

The FAR-EDGE DDA services span the Edge, Ledger and Cloud Tiers of the FAR-EDGE RA, as illustrated in the following paragraphs.

A. Overview of DDA Tiers

Based on the principles of the FAR-EDGE RA, we have implemented a Distributed Data Analytics (DDA) platform, which enables integrators of factory automation solutions to specify and implement highly distributed data analytics logic, based on data stemming from different parts of a plant. DDA is classified as a reusable, self-sustained component (i.e., "enabler"), which supports the functionalities of the Analytics Domain of the FAR-EDGE RA. The DDA platform implementation spans both the Edge and the Ledger Tiers of the FAR-EDGE RA:

- The Edge Tier that provides the means for accessing and routing field data. Moreover, at the Edge Tier the Edge Analytics Engine (EAE) engine is implemented, which provides the means for executing locally scoped data analytics functionalities, and
- The Ledger Tier leverages "Smart Contracts" that manage analytics configurations. A Smart Contract keeps track and synchronizes information across multiple Edge Gateway nodes. In this way, it provides the means for executing factory-wide data analytics, which span multiple locally scoped analytics functions running in Edge Gateways.

Moreover, the DDA implementation takes advantage of the Cloud Tier as well, where plant-wide data are collected, aggregated and consolidated.

In this section, we present the specification and implementation of the Edge Tier of the DDA platform, which is configurable with almost zero programming. Likewise, the next section illustrates the implementation of the Ledger Services that support the Ledger Tier of the DDA platform.

B. DDA's Edge Tier: The Edge Analytics Engine (EAE)

The EAE is a runtime environment hosted in an EG, i.e., at the edge of an industrial automation deployment. It is the programmable and configurable environment that executes data analytics logic locally in order to meeting stringent performance requirements, mainly in terms of latency. While the Ledger Services are in charge of managing Smart Contracts and executing distributed analytics across EGs, the EAE is in charge of data analytics within a single EG. The EAE is also configurable, while comprising multiple analytics instances that are driven by multiple smart contracts. It consists of the following main components:

- the EA-Orchestrator;
- the EA-Processor;
- the Local EA-Repository,

which are described in following paragraphs.

The **EA-Orchestrator** provides the run-time environment that controls and executes EA instances, which are specified in a format that is conveniently called Analytics Manifest (AM). In particular, the EA-Orchestrator is able to parse and execute analytics functions and rules specified in an AM. The following statements define the EA-Orchestrator main operation:

- An AM defines a set of EA functionalities, as a graph of processing functions, which can be executed by the EA-Processor.
- The EA-Orchestrator parses an AM and executes the analytics functions that they comprise.
- The EA-Orchestrator is able to execute multiple, concurrent analytics instances. The latter are specified in AMs.

From an implementation perspective, AMs are represented in different forms such as: a configuration file or an entry in a database, or even a part of a smart contract in the blockchain. No matter the implementation technology, the semantics of the AM specify an analytics instance. Hence, the underlying mechanisms that support execution of AMs are independent from specific implementation technologies, as they are based on the implementation agnostic file format that is available as part of the open source implementation of the EAE.

The AM includes the information needed to drive the operation of the EA-Orchestration, including for example the attributes and sequences needed to setup the required jobs on the EA-Processor. As part of its operation the EA-Orchestrator MAY instantiate multiple EA-Processor instances for the purposes of executing an EA instance,

which is described through an AM. Each AM holds the attributes and sequences to set up the required processor jobs in order to serve one EA instance (i.e., one AM).

The EA-Processor implements the data processing functionalities that are necessary to implement an EA task. These functionalities are encapsulated in different processor types, including:

- **Pre-processors,** which prepare data streams for analysis, based on the specifications of the target analytics tasks. A pre-processor interacts with a Data Bus in order to acquire streaming data from the field. At the same time, it also produces and registers new streams in the same Data Bus.
- **Analytics Processors**, which apply analytics algorithms to one or more data streams. Similar to the pre-processor, the analytics processor consumes and produces data through interaction with the Data Bus.
- **Store Processors**, which are used to store streams to repositories.

Pre-processors, analytics processors and store processors define three different types of functionalities that are supported by the EAE. Given these processor types, a specific instance of EA is implemented by setting up multiple processors, which are connected in a graph-like fashion thus forming a topology. The topology is specified in the AM, which will be represented as a Smart Contract. The topology and the overall process are controlled by the EA orchestrator.

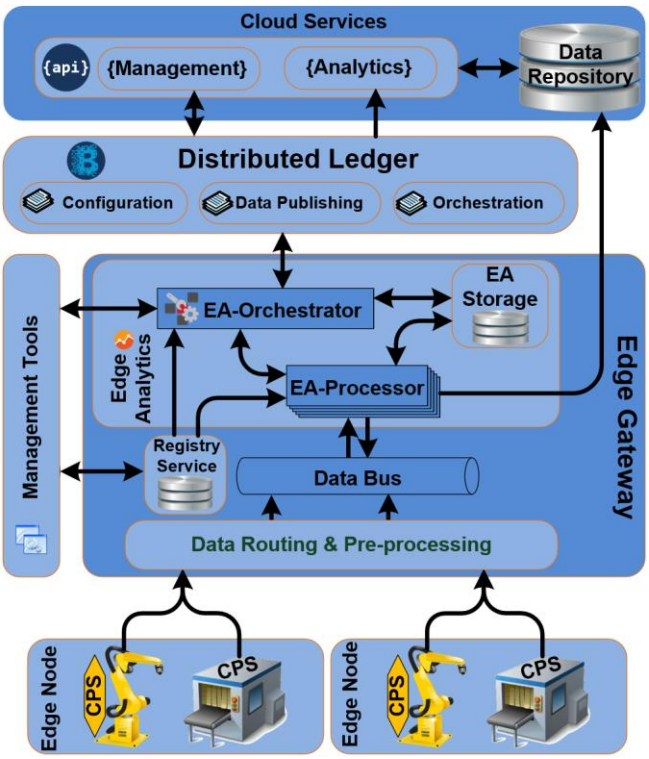

Figure 4. Anatomy of the Edge Analytics Engine

Figure 5 illustrates an example topology and runtime operations for EA Processor. In this example, two streams (CPS1 and CPS2) are pre-processed from Processor Job 1 (i.e., Pre-Processor) and Processor Job 2 (i.e., Pre-Processor) equivalently in order for an analytics algorithm (i.e., Processor Job 3) (i.e., Analytics Processor) to be applied to them. Finally, the result needs to be stored to a Data Storage with the help of Processor Job 4 (i.e., Storage Processor). The setup and runtime operation of the EA-Processor entails the following steps:

- **Step1 (Set-up)**: Based on the description of the topology and required processors in the AM, the EA-Orchestrator instantiates and configures the required Processor jobs.
- **Step2 (Runtime)**: Processor Job 1 consumes and pre-processes streams coming from CPS1. Likewise, Processor Job 2 consumes and pre-processes streams coming from CPS2.
- **Step3 (Runtime)**: Analytics Processor Job 3 consumes the produced streams from Processor Job 1 and 2 for applying the analytics algorithm.
- **Step4 (Runtime)**: Store Processor Job 4 consumes the data stream produced from Processor Job 3 and forwards it to the Data Storage.
- **Step5 (Runtime)**: Data Storage persists the Data coming from Store Processor Job 4.

Beyond this simple example, much more complex EA workflows can be implemented based on combination of the three different types of processors. The supported scenarios are only limited by the expressiveness of the domain specific language / format that is used to define and represent an AM.

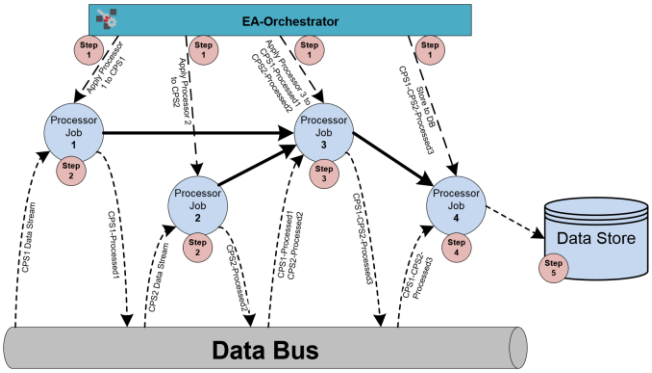

Figure 5. EA Topology Example

C. *Using the EAE for Edge Analytics*

There are two main ways in which solution developers and integrators can use the EAE:

- **Configuration and execution of analytics queries**: First, they can configure and formulate an analytics query, while they can accordingly execute based on the EAE runtime.
- **Extension of the EAE with analytics capabilities**: Second, they can extend the EAE enabler with additional processing capabilities, which respecting the structure and specification of the engine.

These two ways for taking advantage of the EAE are illustrated in the following paragraphs.

In terms of the configuration and Runtime Execution of Analytics Queries, integrators can take advantage of the EAE API in order to configure and execute analytics queries within an EG. The process includes the following steps:

- **Discovery of Devices**: The first step involves discovery of field devices residing in a devices' registry. Devices define the available data sources to be analyzed by the EAE.
- **Discover available processors**: Following the discovery of devices, available data processors registered in the registry are dynamically identified as well. As already outlined there are three types of processors (i.e., preprocessors, analytics, storage) and multiple instances of each one might be available. Each distinct instance is providing different functionalities based on different implementations.
- **Define and create the Analytics specification:** Based on the available devices and processors, a manufacturer or solution integrator can specify an AM, which defines their desired EA tasks. The definition of the AM comprises a flow of processors, including processor of all three types (i.e., pre-processing, analytics, storage) supported by the EAE engine. It also defines the analytics results to be produced, as well as where they are to be stored / persisted. The specification of the AM can take place based on the use of the EAE's RESTful API. However, in future releases of the EAE we plan to provide a GUI tool in order to facilitate zero-programming specification of the EA tasks**.**
- **Execute the AM at runtime**: This step involves the runtime execution of the AM through the EA-Orchestrator using its API. With the AM at hand, this step is straightforward and involves the loading an execution of the specification of the manifest. Upon the AM's execution, the analytics results are produced in the forms of name/value pairs, which are stored as specified by the StoreProcessor.

In terms of extending the EAE with Processing & Analytics Capabilities, AMs can be configured and used. AMs provide a convenient mechanism for defining and executing analytics based on a set of available devices and processors. Integrators are able to extend the analytics capabilities of the EAE, based on the specification and deployment of additional processing functions. Additional processing functions have to be of one of the specified types, which will allow their integration and use within AMs.

The process of extended EAE's capabilities involves the following steps:

- **Implementation of a Processor Interface**: In order to extend the EAE with a new processor, an integrator has to provide an implementation of a specific interface, i.e., the interface of the processor. In practice, each of the three processor types comes with its own interface, which specifies its behavior in the scope of the EAE engine.
- **Registration of the Processor to the Registry**: Once a new processor is implemented, it has to become registered to the registry. This will render it discoverable by solution developers and manufacturers that develop AMs for their needs, based on available devices and processors.
- **Using the processor**: Once a processor becomes available, it can be used for constructing AMs.

D. EAE Open Source Implementation

Apart from a detailed specification of EAE in terms of interfaces, APIs and data schemas for the various processors and the AMs, we have also implemented a prototype of the EAE as open source software [15]. The structure of the implemented system is depicted in Figure 6. As evident in the figure, we take advantage of a Docker container for each distinctive component of our deployment in order to facilitate the distribution, integration and scalability of the system. The Data Bus of the implementation is based on the Apache Kafka platform, which is a distributed system that scales out easily, while offering very high throughput for both publishing and subscribing tasks. Moreover, Kafka supports multi-subscribers and automatically balances the consumers during failures.

The EA-Orchestrator component is also deployed in a Docker container. Hence, the EA-Orchestrator API can be invoked from third party RESTful Client Application (i.e., Postman). To this end, a postman script mapping to the Orchestrator API is offered from GitHub. At the same time, predefined test scripts (i.e., scripts corresponding to AM manifests) have been generated with known actors (CPSs, EA-Processors, configuration attributes etc.).

The EA-Processor component is also deployed in a Docker container. It subscribes to the Data Bus based on the known device IDs. The EA-Processor operates based on a known Number and types of Data Streams. It leverages a static data format.

All available processor types can be used in order to provide a complete test environment including the pre-processing, analytics and analytics storage processors.

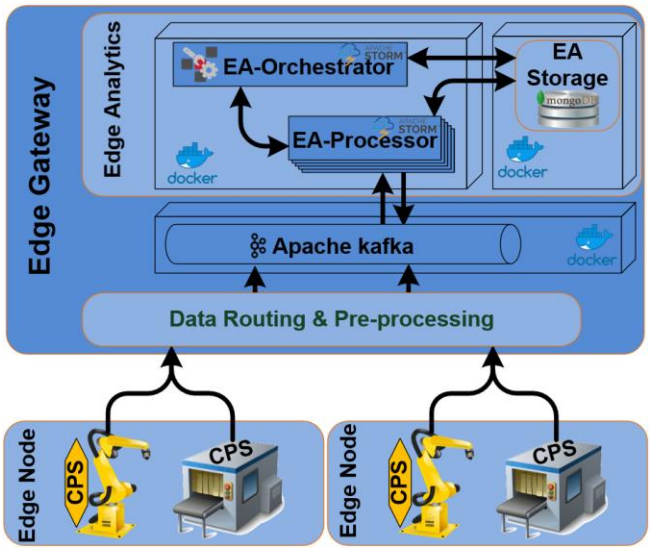

Figure 6. Edge Analytics Engine Implementation

V. LEDGER SERVICES FOR FACTORY WIDE DISTRIBUTED DATA ANALYTICS

The FAR-EDGE Ledger Services enable the most innovative part of the DDA platform, as illustrated in the following paragraphs.

A. Overview

The DDA Platform uses Ledger Services in order to configure plant- and factory-wide analytics processes. Each configuration of analytics algorithms maps to a specific Ledger Service. Every Ledger Service configures one or (usually) more analytics instances. The underlying Distributed Ledger keeps track of multiple analytics configurations. Such configurations are executed by the DDA on production processes that run simultaneously in various locations of the factory.

Moreover, when one analytics task spans multiple EAE instances, a Ledger Service is used to collect local results and implement aggregating logic.

B. Implementation Considerations and Baseline DLT

As explained in the FAR-EDGE RA, the Ledger Tier and Ledger Services are based on DLT – i.e., a Blockchain platform. Concretely, this platform is the Hyperledger Fabric (HLF), which is a commercial-grade Blockchain implementation. HLF has been selected for a number of reasons including its business-friendly open source license, its larger and active community, as well as its support for custom transaction logic (i.e., "smart contracts") and custom data models. Moreover, HLF is a "permissioned" Blockchain as it supports private networks, which are the primary choice for industrial automation deployments.

The HLF architecture is illustrated in Figure 6. Membership and Orderer are the two elements of the system that are not decentralized, being implemented as central services. Peers, on the contrary, are an arbitrary number of computing nodes that can be deployed anywhere – typically on Edge Gateways – and that run in parallel, providing all the basic services that support the lifecycle of ledger transactions: validation (Endorser), confirmation (Committer), state persistence (Ledger) and listener notification (Events). Last but not least, Peer nodes are where Ledger Services are deployed and run.

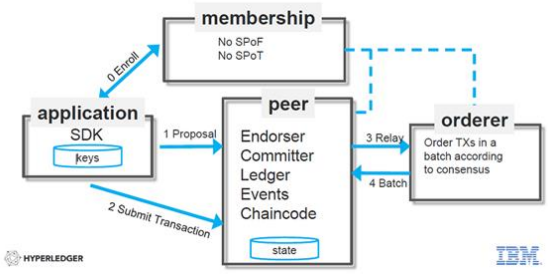

Figure 7. Hyperledger Fabric Logical Architecture [14]

C. Ledger Services

At the platform level, a Ledger Service is a *Chain code* program – i.e., the HLF-specific term for a smart contract. It is designed to support a well-defined, application-specific process. In particular, it is responsible for defining a data model, executing business logic and enforcing access and usage policies. The state of the process is automatically maintained and persisted in the background by the HLF platform, which logs every state change in a distributed ledger that is replicated across all peer nodes.

The data model is shaped by the Chain code itself: a dedicated data store is allocated and initialized by a special code section when the Chain code is first deployed. Once the data store is initialized, no structural changes are expected to happen. It is worth noting that Chaincode instances – and their related data store – are deployed on all peer nodes simultaneously.

Application logic is also coded in the Chain code and is delivered as a number of service endpoints that can be called by clients over the network. These endpoints represent the API of the Ledger Service: only through them callers can query and change its state. The API can be invoked by authorized clients following some well-documented calling conventions of the HLF platform. State-changing calls are managed as a "transaction" by the platform: if the call executes successfully, changes are applied to the persistent storage in all peer nodes; on the other hand, if any error condition is detected (e.g., the Chaincode raising an exception), the platform guarantees that any partial change is reverted.

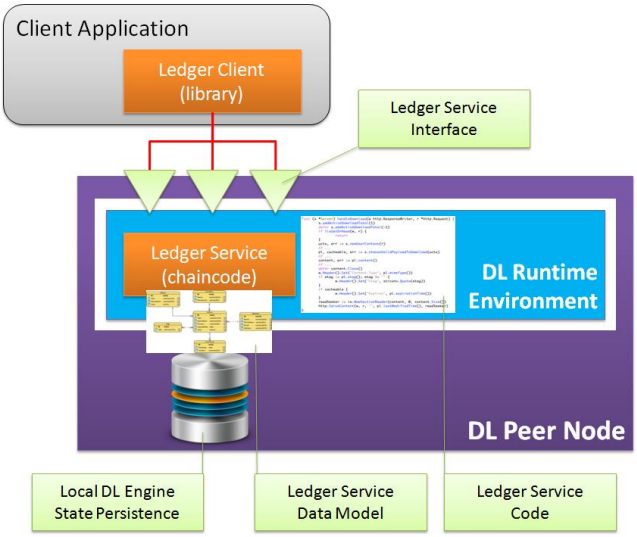

Figure 8. Ledger Services Architecture

In the DDA context, each Ledger Service comes with its own client software library: the Ledger Client. The library provides an in-process API (e.g., Java classes and methods) the matches the network API from a functional point of view but has a much simpler call semantics and hides a lot of HLF-specific technicalities (e.g., user authentication through digital certificates). Ledger Clients can be embedded into

client applications at design time, and used at runtime as a local proxy of the actual Ledger Service API. Figure 8 illustrates the concepts described above from an architectural perspective, focusing on a single peer node.

As already outlined, peer nodes are autonomous sub-systems that run in parallel to provide decentralization and redundancy: each one holds a synchronized copy of the distributed ledger (i.e., the global state of all Ledger Services plus the full history of state-changing transactions) and executes code Ledger Services inside a sandbox environment that isolates each of them from all the others. In order to dynamically adapt the system to the changing needs of the shopfloor, peer nodes can be added to or removed from the running system without any downtime. A number of peer-to-peer protocols are used by peer nodes to collaborate seamlessly with each other, so that the whole system appears to its users as being monolithic. In Figure 9, this relationship is depicted by the "DL Protocols" logical block, that represents the use of common standards for inter-peer communication. Applications can link to the Ledger Service API they are interested in on any peer node of their choice, as all nodes are identical: a service call results in the same code being executed in parallel on each and every node. This redundancy mechanism is what makes the DL a truly decentralized system with exceptional scalability, trustworthiness and reliability properties.

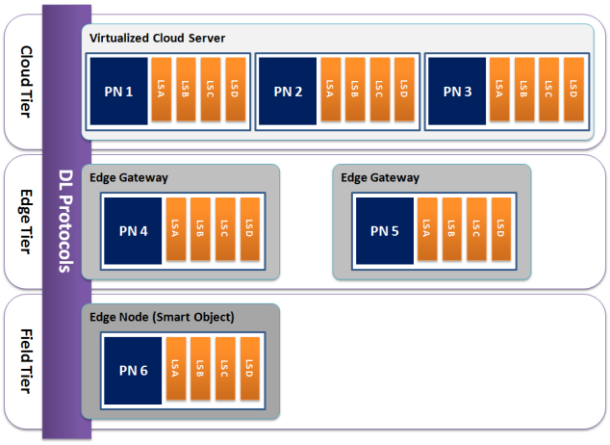

Figure 9. Distributed Ledger Protocol operating across Edge Gateways

In the context of the DDA platform, peer nodes are usually – but not mandatorily – installed on Edge Gateway servers, together with Edge Tier components. This setup allows for clients that run on Edge Gateways, like the EAE, to refer to a local address by default when resolving Ledger Service endpoints. However, peer nodes can be as easily deployed and used on the Cloud Tier, to make them addressable from anywhere; or even embedded into Smart Objects on the Field Tier, to turn the Smart Objects into members of a collaborative P2P (Peer-to-Peer) network.

The definition of access control policies for Ledger Services, and their enforcement at runtime, are built-in features of HLF. There is a fair degree of flexibility in the HLF security subsystem, as individual service endpoints can be optionally protected my means of attribute-based access control (ABAC). At the most basic level, though, all nodes of the network – including the clients – must have a strong digital identity and be authorized by a central authority in order to join the system. On the other hand, when application-specific control is required, the Ledger Service can manage it as part of the implementation.

D. *Self-Adjustment and Recofiguration (SAR) Service*

Self-Adjustment and Reconfiguration (SAR) is an infrastructural feature of the DDA platform. It supports the capability of Smart Objects on the shopfloor to join & leave the system autonomously and to adapt themselves to changing needs and environments in a coordinated way. SAR exploits features of the Ledger Tier, in particular those related to the decentralized coordination of local processes.

The SAR architecture follows the FAR-EDGE RA, spanning three of its layers. The bottom one is the Field Tier, populated by Edge Nodes (EN); right above it, the Edge Tier where a number of Edge Gateways (EG) run some Data Routing components; on top, the Ledger Tier hosting a dedicated Ledger Service: the SAR Service. This design, represented in Figure 10, is driven by a central concept of the FAR-EDGE RA, which breaks down globally-scoped systems into "local clouds". In the SAR context, Data Routing components on EGs act as "caching proxies" of the SAR Service. More specifically, each EG runs a local device registry that is actually partial view over the master one maintained by the SAR Service. The objective of this design is to allow a local cloud, composed by one EG and a number of EN satellites, to act as a modular unit which can be plugged in and out, and even keep working when temporarily disconnected from the main factory network.

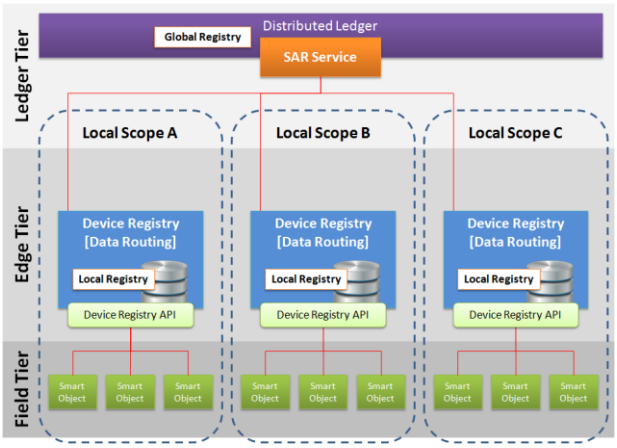

Figure 10. SAR Service Overview

The SAR Service enables the registration, discovery and de-registration of devices that are producers or consumers of *data streams* – i.e., live data flowing from the shopfloor that must be processed in real time. Devices can be either *real* or *virtual*. Real devices can be Smart Objects having the built-in capability of registering and de-registering themselves according to needs (as depicted in Figure 10 above), or

passive IoT sensors that need an administrator to perform these tasks manually. Virtual devices are, instead, computing processes that run on some network node. An example of a virtual data consumer is an analytics program that runs on an EG machine. A virtual data producer may be a program that extracts live data from a legacy database and streams it using some IoT protocol.

The SAR Service also provides endpoints for the creation and decommissioning of communication channels between data producers and consumers. Channels are an abstract notion used to govern how data consumers can connect to data producers, and are the foundation for the enforcement of a device-level access control mechanism. An example use case can help illustrate this point: when a given data consumer wants to establish a new connection to a known data producer (presumably discovered using the registry), it will first need to obtain the authorization to do so from the infrastructure. This is done by means of a SAR Service API call: if successful, this call creates a channel descriptor into the registry. The data producer will then be able to check if incoming connection requests come from "authorized" consumers, and refuse to service them if not.

VI. PILOT PLANT DEPLOYEMENT

We have deployed, tested and demonstrated the DDA in the scope of a pilot plant, which has been built in the scope of the Technology-Initiative Smart Factory-KL. The latter is a testbed for testing and demonstrating the future factory of industrial automation. The plant is arbitrarily modifiable and expandable (flexible), connects arbitrary components of multiple manufacturers (networked), enables its components to perform context-related tasks autonomously (self organizing) and emphasizes user friendliness (user-oriented).

The testbed comprises three Infrastructure Boxes (IB). Each IB comprises energy sensors, which are accessible via an MQTT interface. Energy data are provided every second and comprises information such as the total real power, the total reactive power, the total apparent power, the total real energy, the total reactive energy, the total apparent energy and more. As part of the DDA deployment, we provide the means for computing the hourly daily consumption of the real power and the real energy for each IB and for all three IBs. To this end, on Edge Gateway (comprising an EAE) has been deployed in each one of the IBs.

A data model comprising a Data Interface (DI), a Data Source (DSD) and a Data Kind (DK) has been developed and used to generate a Data Source Manifest (DSM), which is registered in each Edge Gateway. In-line with specification of the EAE, a number of processors have been modelled and developed, including a processor for hourly average calculation from a single data stream, as well as a processor for persisting results in a MongoDB.

The specified at models are used to generate the Analytics Processor Manifest (APM) for each required processor, which is registered to the Edge Gateway. Instances of the above listed processors are created in order to calculate hourly averages from the total real power and from the total real energy data streams. Moreover, the processor for persisting results is instantiated in order to store results at the edge tier (i.e., in the Edge Gateway's MongoDB) and at the cloud tier (i.e., a cloud-based MongoDB destined to store global results). The former (edge tier MongoDB) holds the results of EA, while the latter (cloud tier MongoDB) holds the results of factory-wide DDA. Further deployments will be made to get the data from individual Smart Factory-KL modules. These modules can provide additional data such as presence of other nearby modules, current status of the production, state of the module, the order that is being processed along with its priority and other attributes.

Ledger Services are used for orchestrating the instantiated processors. The orchestration is based on an AM, which is registered and controlled through the distributed data Analytics Engine API.

VII. CONCLUSIONS

This paper has introduced a novel RA for decentralized industrial automation, which combines the benefits of edge computing (i.e., near real-time control and data processing) with the capabilities of blockchains in terms of synchronizing distributed processes in scalable way. We have also illustrated a tangible implementation of a DDA platform, which adheres to the main principles of the presented RA. In particular, the DDA platform is empowered by a runtime time environment for programmable, high-performance EA, as well as by a set of distributed ledger services, which enable secure state sharing across multiple analytics processes. The main innovation of the edge tier implementation lies in the fact that it provides the means for specifying, configuring and executing analytics functions with minimal programming. At the same time, the innovation of the ledger services lies in the pioneering use of a permissioned blockchain for synchronizing distributed processes.

The implemented DDA platform and its main enablers are available as open source software [15] [16], which represents one of our tangible contributions to the growing community of Industry4.0 and Industrial IoT researchers and engineers. We have also already deployed a concrete analytics use case in a pilot plant. Our vision and implementation roadmap includes benchmarking the performance of our blockchain system against industry requirements. Based on this benchmarking, we plan to provide to the Industry4.0 community concrete insights on the scope and the limitations of DLT technology for industrial automation and analytics applications.

ACKNOWLEDGMENT

This work has been carried out in the scope of the FAR-EDGE project (H2020-703094). The authors acknowledge help and contributions from all partners of the project.

REFERENCES

[1] M. Isaja, J. Soldatos, and V. Gezer, “Combining Edge Computing and Blockchains for Flexibility and Performance in Industrial Automation,” In the Proc. Of the The Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM 2017, November 12 - 16, 2017 - Barcelona, Spain.

[2] P. Vrba et al., Review of Industrial Applications of Multi-agent Technologies," Service Orientation in Holonic and Multi Agent Manufacturing and Robotics, Studies in Computational Intelligence Vol. 472, Springer, pp 327-338, 2013.

[3] P. Leitão, “Agent-based distributed manufacturing control: A state-of-the-art survey,” Engineering Applications of Artificial Intelligence, vol. 22, no. 7, pp. 979-991, Oct. 2009.

[4] F. Jammes and H. Smit, “Service-Oriented Paradigms in Industrial Automation Industrial Informatics,” IEEE Transactions on, pp. 62 – 70, vol. 1, issue 1, Feb, 2005.

[5] T. Cucinotta and Coll, “A Real-Time Service-Oriented Architecture for Industrial Automation,” Industrial Informatics, IEEE Transactions on, vol. 5, issue 3, pp. 267 – 277, Aug. 2009.

[6] W. Shi, J. Cao, Q. Zhang, and Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016. doi: 10.1109/JIOT.2016.2579198.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” Proceedings of the first edition of the MCC workshop on Mobile cloud computing, MCC '12, pp 13-16.

[8] H2020-703094 FAR-EDGE Project 2018. [Onine], Available from: http://www.far-edge.eu 2018.05.30

[9] Industrial Internet Consortium. 2017. The Industrial Internet of Things Volume G1: Reference Architecture, version 1.8. (2017). [Online], Available from: http://www.iiconsortium.org/IIRA.htm 2018.05.30

[10] Industrial Internet Consortium. IIC Edge Intelligence Testbed. 2017. [Online], Available from: http://www.iiconsortium.org/edge-intelligence.htm 2018.05.30

[11] EdgeX Foundry Framewok 2017. [Online], Available from: https://www.edgexfoundry.org/ 2018.05.30

[12] K. Schweichhart. “Reference Architectural Model Industrie 4.0 - An Introduction,” April 2016, [Online], Available from: https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf 2018.05.30

[13] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, and B. Amaba, "Blockchain technology innovations," IEEE Technology & Engineering Management Conference (TEMSCON), Santa Clara, CA, 2017, pp. 137-141.

[14] A. Le Hors and R. Strukhoff, “Hyperledger Fabric Approaches v1.0 with Better Scalability and Security,” November 2016. [Online], Available from: https://www.altoros.com/blog/hyperledger-approaches-version-1-0-with-better-scalability-and-security 2018.05.30

[15] FAR-EDGE, EdgeAnalyticsEngine 2018. [Online]. Available from: https://github.com/far-edge/distributed-data-analytics 2018.05.30

[16] FAR-EDGE, DistributedLedger 2018. [Online]. Available from: https://github.com/far-edge/DistributedLedger 2018.05.30