IoTwins: Design and Implementation of a Platform for the Management of Digital Twins in Industrial Scenarios

Andrea Borghesi[a], Giuseppe Di Modica[a], Paolo Bellavista[a], Varun Gowtham[b], Alexander Willner[b], Daniel Nehls[b], Florian Kintzler[c], Stephan Cejka[c], Simone Rossi Tisbeni[d], Alessandro Costantini[d], Matteo Galletti[d], Marica Antonacci[e], Jean Christian Ahouangonou[f]
[a]*DISI, University of Bologna,* [b]*Fraunhofer FOKUS / TU Berlin,* [c]*Siemens AG Austria, Vienna*
[d]*INFN-CNAF, Bologna,* [e]*INFN, Bari,* [f]*ESI GROUP, Rungis*

***Abstract*—With the increase of the volume of data produced by IoT devices, there is a growing demand of applications capable of elaborating data anywhere along the IoT-to-Cloud path (Edge/Fog). In industrial environments, strict real-time constraints require computation to run as close to the data origin as possible (e.g., IoT Gateway or Edge nodes), whilst batch-wise tasks such as Big Data analytics and Machine Learning model training are advised to run on the Cloud, where computing resources are abundant. The H2020 IoTwins project leverages the digital twin concept to implement virtual representation of physical assets (e.g., machine parts, machines, production/control processes) and deliver a software platform that will help enterprises, and in particular SMEs, to build highly innovative, AI-based services that exploit the potential of IoT/Edge/Cloud computing paradigms. In this paper, we discuss the design principles of the IoTwins reference architecture, delving into technical details of its components and offered functionalities, and propose an exemplary software implementation.**

1. Introduction

Big data has become a fundamental game changer in the industrial and service sector over the last few years, but only recently there has been a significant shift of focus from the hype surrounding it to finding real value in its use. Modern data analytics, artificial intelligence (AI) and machine learning (ML) techniques have an unprecedented chance to bring companies of products and services into the world of digital business, but a number of barriers have to be reduced.

Developing intelligent systems requires mastering complex and rapidly evolving tools and techniques, introducing substantial delays and costs in product/process design, deployment, test, and refinement. Recent advances based on deep learning require access to very large sources of curated data, as well as significant computational resources for training. The runtime execution and online refinement of learned models, in particular in industrial environments, often need to be at the premises of the systems generating the big data, e.g., to locally monitor, control, and adapt the components of a manufacturing production line under tight latency and reliability requirements, while preserving an adequate degree of data privacy. Deployment and exploitation of new forms of systems and services require substantial investments in infrastructure at the server side (where relevant cloud resources are often needed for model learning and simulation), at the edge side (e.g., to extend manufacturing machinery and their gateways on the industry plant premises with edge computing functionality), and also in terms of integration efforts.

The above investments, which could be perceived as cost-prohibitive, are now becoming a business necessity for innovative companies, in both the manufacturing and facility management domains, with particular attention to the rich ecosystem of EU small and medium enterprises (SMEs). The H2020 IoTwins project aims to lower the barriers for building edge-enabled and cloud-assisted intelligent systems and services based on big data for the domains of manufacturing and facility management, by harmonising standards to enable interoperability, and by developing an easy-to-use service layer that facilitates and decreases the cost of integration and deployment.

IoTwins aims to design a framework for a seamless, straightforward and loose integration of already developed and deployed industrial software components running in typical long-lived industrial test-beds. IoTwins proposes a reference architecture that offers common tools (data shift, code migration, software components pipelining) to consolidated industrial testbeds that need to evolve over time. The design of the IoTwins architecture envisioned the participation and active collaboration of project partners running industrial manufacturing and facility management business, and wishing to enhance and evolve their industrial testbeds in a sustainable manner. They contributed by providing needs and technological requirements that eventually guided the definition of a distributed, digital-twins based platform leveraging both the cloud and the edge.

In this paper, we discuss the design principles of the IoTwins reference architecture. In particular, we delve into the details of the requirements elicitation process and the functionalities provided by the layered architecture. Finally, we present an exemplary software platform that adheres the architecture's design principles. The paper is structured as follows. In Section 2, a survey of related word is presented.

In Section 3, the design principles of the IoTwins reference architecture are thoroughly discussed. Section 4 concludes the work and touches on future work.

2. Related work

In the transition to Industry 4.0 based systems, it is by design necessary to switch to digital approaches such that the operational technology of manufacturing industries embrace software defined approaches [4]. A key concept that aids this transition is digital twins, which are virtual representations of physical assets [5, 10]. Digital twins not only facilitate administration of physical assets but also aid in modelling the physical characteristics of the asset, so that the digital representation can be transferred or replicated on computing nodes [11]. In a software system, it is important to enable deployment and maintenance of digital twins [6, 8, 14]. The edge computing paradigm has further closed the gap of application of digital twin concepts in manufacturing industries [2, 3, 12, 15].

The European Commission has funded several project/activities under the umbrella of H2020. Some of the most relevant are listed: eXtreme-DataCloud[1] envisions the development of cloud technologies which are open and inter-operable. deep-Hybrid-DataCloud[2] aims at bridging cloud and intensive computing resources to explore large datasets for artificial intelligence, deep, and machine learning. Collaborative environments and efforts to streamline the application of digital twins are focus for bodies such as, SPARTA[3] in field of cybersecurity, AI4EU[4] unifying the European Union infrastructure and framework for AI advances and Fortissimo[5], a collaborative project that enabled European SMEs to be more competitive globally through the use of advanced modelling. Edge and fog computing was the focus of projects such as AUTOWARE[6] bringing fog technlogy into manufacturing industry, BEinCPPS[7] enables real-time machine to machine communication and Boost 4.0[8] is a European initiative that applies big data in manufacturing applications through fog/edge technology. The LarGo! project[9], funded outside the EU network, is a cooperative transnational research project focusing on the large scale rollout of smart grid applications [7].

Orchestration is a key element in the the context of digital twins. Several standardization bodies are responsible in drafting standards to facilitate interoperability and accelerate the incorporation of orchestration and management of digital twins into practice. Some of the notable bodies are listed: The MEC is often seen as the key enabler for offering ultra-low latency and high-bandwidth in edge solutions. ETSI Management and Orchestration (MANO)[10] serves as a standard for management and orchestration of virtual network functions. The Industrial Internet Consortium and OpenFog incorporate orchestration in their reference architectures.

Digital twins have also been addressed by the major cloud providers, e.g., Amazon Web Services (AWS)[11], Microsoft Azure[12] and Google Cloud Platform (GCP)[13] included in their IoT solutions. For example, within AWS IoT, JSON representations of the real "thing" are used as devices shadows, which basically contain a *desired* and a *reported* state. Whenever the device reconnects to the cloud, the reported state is updated with the current state of the device and the delta between the reported and the desired state is propagated to the device to be handled accordingly [1]. Cloud service can work with the most recent reported state and can update the desired state if necessary.

Cloud-edge architectures can be created by using AWS IoT Greengrass[14]. Smaller devices may not connect directly to the cloud, but to a dedicated Greengrass Core (GGC) device located on the edge, able to act locally on the sensed data [1]. In this architecture, the GGC is the only device that needs a connection to the cloud. AWS IoT SiteWise[15] is a managed service provided to collect, store, organize, and monitor data from industrial facilities at large scale. Models of the physical assets, processes and facilities are used to provide metrics, e.g., for the prediction of maintenance issues. Specialized SaaS offerings for Digital Twin functionalities are currently available only on Azure with the Azure Digital Twins[16] that have just recently reached its general availability.

The IoTwins project propose a highly distributed and *hybrid* digital twin model, which provides for an integration of simulative and data-driven models to feed AI services. To the best of our knowledge, very few papers in the literature have adopted this approach.

3. The IoTwins Platform

Digital twins are models to represent a system (infrastructure/process/machine) along with its performance. In the scope of IoTwins, these models enable the description of the system itself and its dynamics (descriptive or interpretative models), the prediction of its evolution (predictive models), and the optimization of its operation, management and maintenance (prescriptive models). They are used to detect and diagnose anomalies, to determine an optimal set of actions that maximize key performance metrics, to effectively and efficiently enforce on-line quality management of production

1. eXtreme-DataCloud: http://www.extreme-datacloud.eu/
2. deep-Hybrid-DataCloud: https://deep-hybrid-datacloud.eu/
3. SPARTA: https://www.sparta.eu
4. AI4EU: https://www.ai4eu.eu/
5. Fortissimo: https://www.fortissimo-project.eu/
6. AUTOWARE: https://www.autoware-eu.org/
7. BEinCPPS: http://www.beincpps.eu/
8. Bosst4.0: https://boost40.eu/
9. http://www.largo-project.eu/
10. MANO: https://www.etsi.org/technologies/open-source-mano
11. AWS: https://aws.amazon.com/iot/
12. Microsoft Azure: https://azure.microsoft.com/overview/iot/
13. GCP: https://cloud.google.com/solutions/iot
14. AWS IoT: https://aws.amazon.com/greengrass/
15. AWS IoT SiteWise: https://aws.amazon.com/iot-sitewise/
16. Azure Digital Twins: https://azure.microsoft.com/services/digital-twins/

processes under latency and reliability constraints, and to provide predictions for strategic planning to help companies, especially SMEs, to significantly improve their profitability through digitalization, as well as to open up new opportunities for them for the creation of new services and business models.

IoTwins proposes a hierarchical organization and interworking of digital twins modelling manufacturing production plants and facility management deployment environments. In particular, such a hierarchy is composed of three digital twins layers:

- **IoT twins** feature lightweight models of specific components and performing big-data processing and local control for quality management operations (low latency and high reliability).
- **Edge twins** deployed at plant gateways and/or at emerging ETSI Multi-access Edge Computing (MEC) nodes, providing higher level control knobs and orchestrating Internet of Things (IoT) sensors and actuators in a production locality, fostering local optimizations and interoperability.
- **Cloud twins** perform time-consuming and typically off-line parallel simulation and deep-learning, feeding the Edge twin with pre-elaborated predictive models to be efficiently executed at the premises of production plants for monitoring/control/tuning purposes.

The main principles that drive the design of the IoTwins platform are openness and software reusability. The platform proposes itself as an open framework that manages digital twins in the Cloud-to-Things continuum, built on top of open-source software and tools, and allowing third-party software to integrate by utilizing open application programming interfaces (API). In the following, we report on the requirements collected from the industrial partners, the formal definition of the platform's use cases and the components structure of the IoTwins architecture.

3.1. Requirements elicitation

The first step towards the creation of the IoTwins platform consisted in gathering functional and non-functional requirements from the project's industrial partners and properly defining the distributed twin features needed by each of them. To collect these requirements, a questionnaire had been prepared and given to each testbed leader to be answered. Afterward, interviews were conducted to clarify points not entirely covered by the questionnaires and to treat testbed specific issues in more detail. The collected requirements spanned both infrastructural-related issues (ranging over all layers covered by the platform: IoT, edge, and cloud) and the services to develop digital twins (e.g., ML models, optimization, and simulation models).

The key observation concerns the heterogeneity of the involved partners, as some testbed already possess partial solutions and components while others are starting from scratch; there is as well a significant variety in terms of desired services, based on the nature and scope of the testbed activity. This observation imposes the proposed platform *to be open and flexible* to accommodate the different needs and to adapt to already existing environments. Concerning IoT and Edge levels, infrastructure solutions were either available or clearly identified for all testbeds. Often, they are based on custom platforms but composed of common building blocks; for instance, the usage of the MQTT protocol for data transfer at the lower layer of the infrastructure is common to various partners.

Moving up to edge and edge-cloud interface, confidentiality is a critical element – all data should be treated as confidential and should thus be transferred encrypted only. Data rates are in the order of tens of GBs per day (raw, uncompressed), which does not impose a huge concern in terms of storage capacity but underlines the need of reliable (possibly wired) network connectivity – buffering strategies on the edge should be considered to cope with possible network interruptions. At the cloud level (REST) APIs are needed to instantiate resources (which should be accessed through Virtual Private Networks) and interact with services offered; cloud resources should be provided for both Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) paradigms (e.g., PaaS is requested to instantiate ML frameworks, SQL and NoSQL databases, and batch clusters creation on the fly). Hardware resources (CPUs, GPUs, storage, etc.) should not present challenges on cloud level but particular care should given to services requiring graphical access and 3D rendering, as the network bandwidth could become a bottleneck.

With respect to the services, the testbeds have very different needs. However there is a potential commonly shared interest in anomaly detection and fault prediction (model introspection is also welcomed). Due to the heterogeneity of the requested services, the IoTwins platform should provide flexible services which can be composed on more complex pipelines; furthermore, it must be possible to tailor the services to specific testbed needs – clear instructions and support must be granted for this scope. In all testbeds, the edge offers sufficient computing capability to execute simple computations (e.g., inference on live data) but the heavy lifting (e.g., simulation and training of ML models) will necessarily be performed on cloud resources.

3.2. Use cases

The definition of the use cases presented in this section aims to meet the requirements elicited from the testbeds. According to such needs, the IoTwins platform will exhibit functionalities to:

- support data transfer between the three infrastructure levels adhering constraints imposed by the specific computation needs (real-time, non-real-time);
- support several data types and diverse data storage needs (short- and long-term storage);
- support data elaboration both on the fly (streamed-data) and at rest (batch-wise)

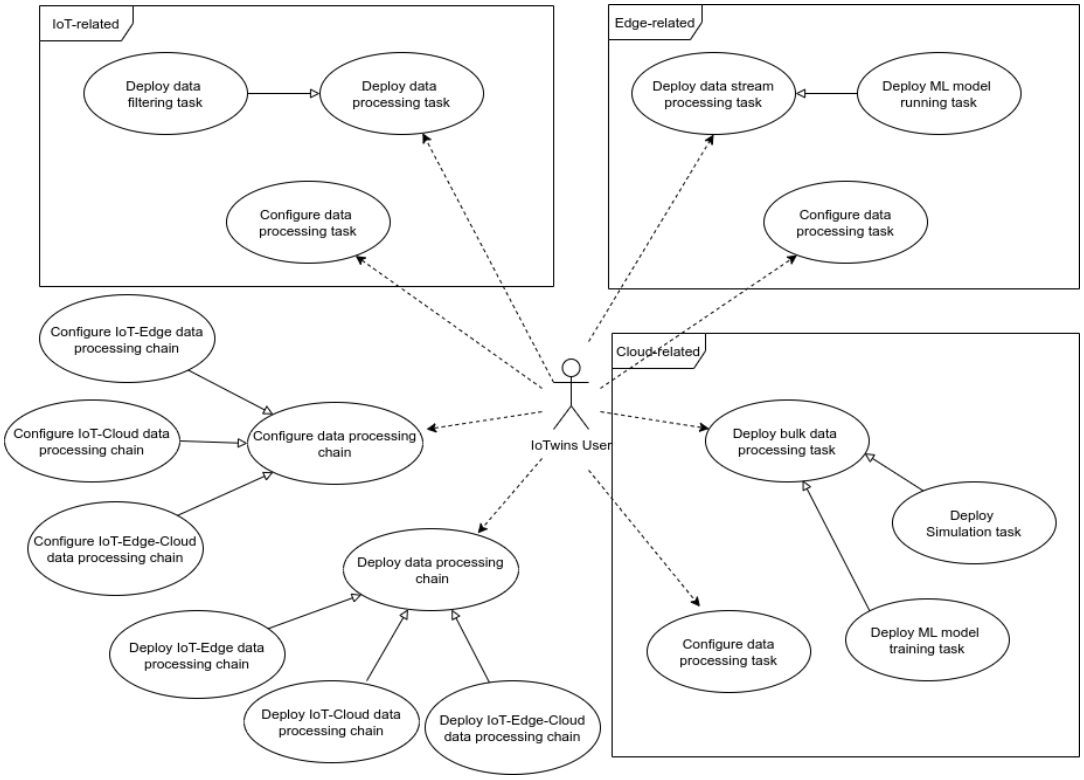

Figure 1: IoTwins platform: Use case diagram

In addition to the administrative services that support deploying, configuring and running services on each computing infrastructure levels (IoT, Edge and Cloud), the platform provides them with services to configure and transparently deploy complete *computing chains*, i.e., chains of software libraries/tools/services that implement the above-mentioned functionalities along the IoT-Edge-Cloud computing continuum.

The *IoTwins User* is the main stakeholder who will benefit from the services offered by the IoTwins platform. In the scope of the project, the IoTwins User role is played by the TBs. Out of the project scope, potential users will be players of manufacturing and facility/infrastructure management sectors that wish to take advantage of the IoTwins platform services to implement digital twin-based business processes. In Figure 1, a UML use case diagram of the *IoTwins User* cases is depicted.

The IoTwins platform enables users to configure, deploy and run data processing tasks on IoT, Edge and Cloud level respectively. These tasks cover all data analytics activities including data filtering, data polishing, data integration, data elaboration, data visualization, and data monitoring. In Figure 1, four specific cases are depicted as specialization of the generic level specific *deploy data processing task* case in each computing level: *Deploy Data filtering task* on IoT level, *Deploy ML model running task* on Edge level and *Deploy ML model training task* and *Deploy Simulation task* on Cloud level respectively. The first one refers to the deployment of a typical light-weight computing task that can be carried out on resource-constrained devices such as the IoT; the second one refers to the deployment of medium-weight computing task to be run on the edge devices; the third and the fourth ones concern the deployment of heavy-weight tasks to be run on powerful resources such as those offered by the cloud.

Along with functionalities to configure and deploy services/tasks at each computing level, use cases have been defined to let users request the activation of multiple tasks distributed along the chain of the computing infrastructures (bottom-left side of Figure 1). Depending on the needs, chains can be configured that span two (IoT-Cloud or IoT-Edge) or all infrastructure levels (IoT-Edge-Cloud). As an explanatory example, a typical data processing chain envisions data sensing and filtering tasks deployed in the IoT, an ML model training task running in the cloud and fed with data at rest, and an ML model execution task running in the edge consuming IoT data streams. Along the chain, data transfer services (bulk data transfer, live data transfer) and storage services (Relational/NoSQL/Time series DB, file-system based storage, etc.) can be instantiated.

3.3. Reference architecture

As large-scale pilots need a solid and flexible e-infrastructure to be effectively implemented, the IoTwins project develops this infrastructure on top of existing data center resources, capable of supporting the needs of the

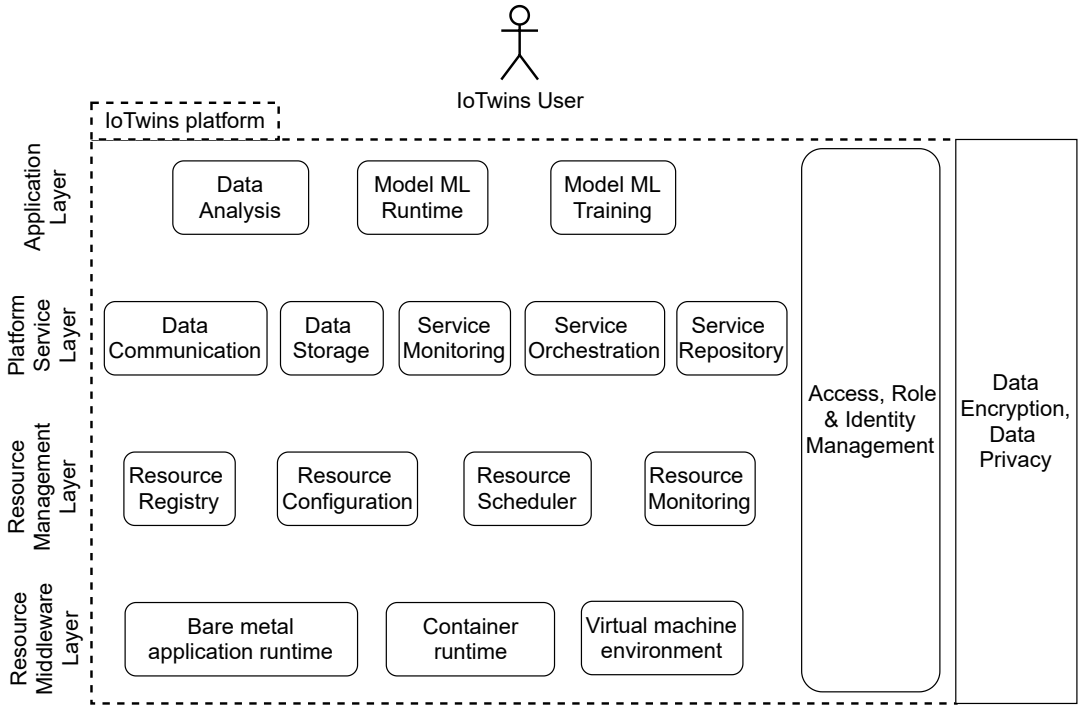

Figure 2: IoTwins platform: Reference architecture

identified use cases. To achieve this goal, the IoTwins architecture is based on a Platform as a Service (PaaS) layer able to leverage disparate hardware resources (traditional commercial clouds, HPC resources offered by research institutions, and on premise industrial private clouds) to process large amounts of data and to exploit the efficient storage technologies/infrastructures made available by the partners. The IoTwins PaaS provides advanced tools for computing and for processing large amounts of data, and to exploit current storage and preservation technologies, with the appropriate mechanisms to ensure security and privacy.

Figure 2 depicts the four-layered IoTwins architecture. Specific technologies and components that implement the depicted functionalities are then selected and used to create the three computing levels IoT, Edge and Cloud respectively. The possibility to implement these functionalities using diverse technologies enables the user to address application and testbed-specific requirements in a flexible way (see Section 3.2).

For the implementation (cf. Section 3.4), components out of a pool of existing (open source or partner-owned) tools were chosen. To show the openness of the architecture, several parts (e.g., the service orchestration and the edge management components) of the reference architecture were realized using multiple competing technologies.

3.3.1. IoTwins platform layers

I. The **Resource Middleware Layer** contains components that implement the virtualization of the underlying computing assets and provide high level functionality to manage these resources. The term *resource* thereby comprises all kinds of virtualized and encapsulated computing resources (virtual machine, virtual storage, container, etc.) ready to use for general-purpose computation. To utilize the power of containerized applications to uniformly operate applications for sensing, controlling, and data filtering to a variety of distributed devices, multiple solutions in addition to the widely used Docker[17] runtime are available. In order to enable usage a multitude of these technologies the containers have to adhere to a certain standard. The project therefore focuses on solutions for Open Containers Initiative[18] (OCI) compatible solutions. This includes Docker[17], Podman[19] and CRI-O[20] on the level of container execution management.

The remote application management functionality on the edge and IoT devices is closely connected to the functionality provided on the Service Provisioning Layer. For example, KubeEdge[21] together with CRI-O[20] is a lightweight solution to enable the remote management of containerized applications on resources by use of a Kubernetes[22] based backend. Kubernetes integration with OpenStack[23] is possible using the Magnum[24] component together with Mesos[25]

17. Docker: https://docker.com
18. OCI: https://opencontainers.org/
19. Podman: https://podman.io
20. CRI-O: https://cri-o.io
21. KubeEdge: https://kubeedge.io/
22. Kubernetes: https://kubernetes.io
23. OpenStack: https://www.openstack.org
24. Magnum: https://wiki.openstack.org/wiki/Magnum
25. Mesos: http://mesos.apache.org

(cf. Section 3.4 for the components used in the reference implementation).

II. The **Resource Management Layer** provides the upper layer with the functionality to manage Resources (i.e., virtualized computing resources) in a transparent way using the following components:

- The *Resource Registry* keeps track of all resources, their features, their availability, etc. Whenever a new resource is available to the platform, a new entry (record) representing that resource will be added to the registry.
- On request the *Resource scheduler* searches for fitting computing resources and schedules tasks on behalf of the requester.
- The *Resource monitor* implements the monitoring of all active computing resources. Resource level utilization will be monitored through a set of KPIs.

III. The **Platform Service Layer** is responsible for the provisioning of IoTwins services. All services are provided using OCI[18]-compliant containers; their images are stored in a container image registry accessible from all computing units. Most of the IoTwins services will be pre-packaged and pre-loaded on the registry, however the system also allows for packaging and upload of customer services (typically, custom applications for data elaboration such as data filtering, simulators, ML models, etc.) during runtime.

A service can belong to one of the two following types:

- *Data Communication*: This category comprises all services that offer support for one-to-one, one-to-many, or many-to-many communication. Support will have to be offered for real-time, bulk data, short-lived and long-lived transfer in general. Example of services providing data transfer support are message brokers based on the publish/subscribe pattern (e.g., Kafka), FTP, NTP, webdav, etc.
- *Data Storage*: These services store data locally on the computing unit where the service has been requested. Data storage services will include relational databases, NoSQL databases, time series DB, file system-based storage, etc.

The *Service Monitoring* component is in charge of monitoring the provisioned services and informing the upper layer about their performance (e.g., end-to-end delay in the case of a data transfer, storage capacity saturation in the case of a data storage service, etc.)

The *Service Orchestrator* component is responsible for assigning IoTwins applications to computing resources. In particular, it is in charge of managing the life-cycle of these applications, starting with the provisioning of all required software components and ending with their disposal. It will monitor applications' health (in terms of QoS) and take corrective actions in case the QoS level is not being sustained as requested. Upon the arrival of an end-user request the Orchestrator will:

a) schedule resources in the computing continuum (IoT $\rightarrow$ Edge $\rightarrow$ Cloud),
b) retrieve the tools/software components/libraries useful to build up the distributed application,
c) install the software components on the target devices, do the necessary pipelining (configuring and adequately connecting components to each other),
d) set up and run the application monitoring infrastructure, and
e) run the applications.

IV. The **Application Layer** is populated by customer applications that perform some kind of elaboration over data. An example on the IoT level could be an application that filters or polishes data before sending it to the edge. On the edge side, there could be an ML model that takes in input data coming from IoT and elaborates actions (e.g., triggering commands to actuators). On the cloud side, there could be a simulator that works on bulk data provided by the edge or the training of an ML model.

Access, Role & Identity Management is a framework of policies and technologies for ensuring that the proper users access the appropriate resources in a computing infrastructure. Those systems not only identify, authenticate, and authorize individuals but also applications and services that can access certain resources.

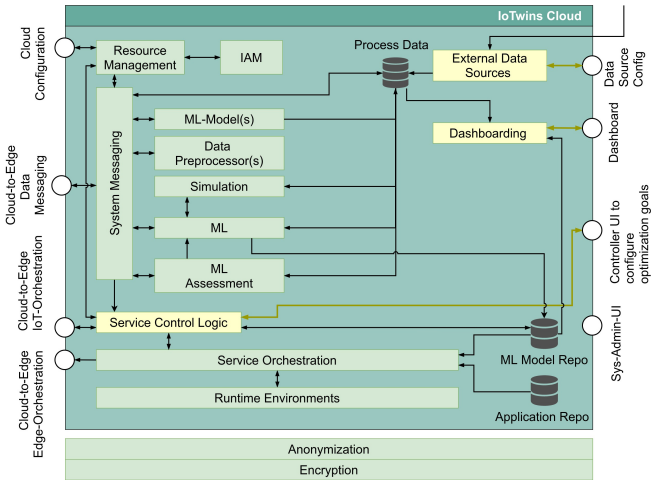

Figure 3: Cloud Level Architecture

3.3.2. Computing levels perspective

The functionalities from the reference architecture (Figure 2) are combined to form systems implementing the following three computing levels:

I. The **Cloud Level** (Figure 3) provides components for central resource management of all architectural levels, i.e., it controls which software is running on which hardware (orchestration) in the complete system. In addition, the Cloud Level stores and provides data from and to the devices on the Edge Level. Third, the Cloud Level provides services to process the data and create (partial) models of the system that can the be used on Cloud Level, Edge Level and/or IoT Level.

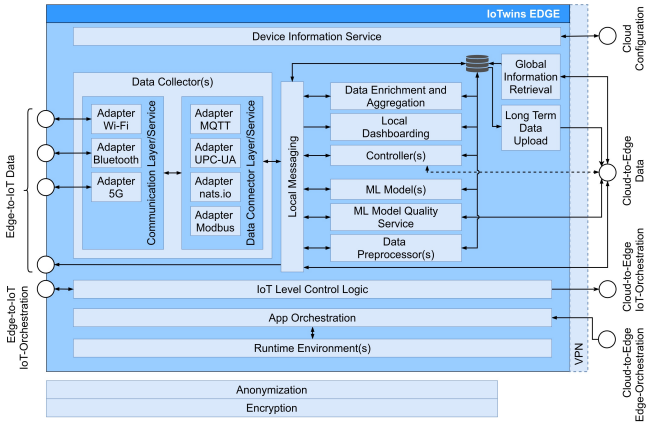

Figure 4: Edge Level Architecture

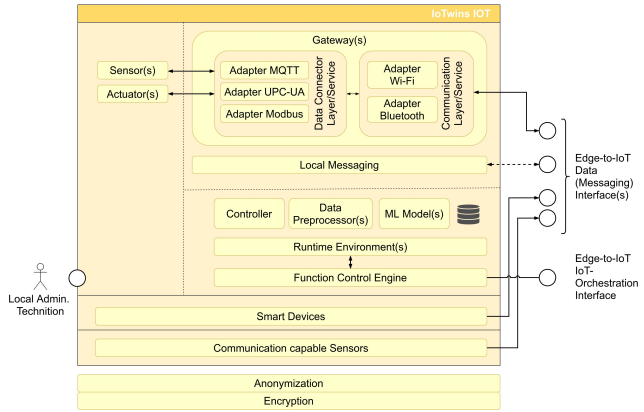

Figure 5: IoT Level Architecture

II. The **Edge Level** (Figure 4) provides components that implement the central orchestration decisions on the edge devices. The orchestration functionality on the Edge Level is also used to run the models that were created on Cloud Level. In addition, the Edge Level devices store and forward sensor data to the Cloud Level and retrieve data and applications from the Cloud Level. The orchestration and execution of machine learning components on the edge devices must thereby be managed with respect to locality (control algorithms are bound to one device) and the available computing resources (optimization and adaptation for hardware properties, e.g., CPU architecture).

III. The **IoT Level** (Figure 5) is the most diverse level. It comprises integrated smart devices that implement all necessary connectivity and pre-processing functionality as well as devices that in their architecture resemble the Edge Level and thus provide methods to orchestrate functionality on these devices.

3.4. Implementation

The IoTwins project proposes a possible implementation of the reference architecture (see subsection 3.3), with a software framework built to provide the functionalities defined by the requirements elicited from the testbeds.

The main component of the framework is INDIGO PaaS Orchestrator [13], a tool for service orchestration acting as resource manager on both the cloud and the Edge layer. This component can instantiate resources on a number of cloud management frameworks (like OpenStack[26], OpenNebula[27]) and Mesos clusters[28]. It takes the deployment requests, formulated via templates complying the TOSCA YAML Simple Profile v1.0 [9], and deploys application components on the best available cloud infrastructure according to SLAs, monitoring needs and other service requirements the cloud provider is able to fulfill.

TOSCA was selected as the language for describing application topologies, thanks to the wide adoption of this standard and the compliance granted by both OpenNebula (through the IM) and OpenStack (through Heat). The INDIGO PaaS layer is able to provide automatic distribution of applications and/or services over a hybrid and heterogeneous set of IaaS, such as the edge nodes and both private and public clouds.

The PaaS layer takes as input the description of a cluster of services/applications and provides the best fitting resources. During this process, the PaaS layer evaluates data distribution, and select resources that are as much close as possible close to the storage services hosting the data serving the specific applications/services.

The orchestration service is integrated with the INDIGO-IAM Service to meet the project requirement of a unified identity and access management. INDIGO-IAM is an open source software that provides a layer where identities, enrollment, group membership and other attributes and authorization policies on distributed resources can be managed in a homogeneous way. It supports identity federations and other authentication mechanisms such as X.509 certificates, social logins, SAML Identity Providers (IdPs) and OpenID Connect providers.

The INDIGO Paas fulfils the IoTwins platform requirements, providing the functionalities of *Service Provisioning* and *Orchestration*, *Resource Management*, and *Access, Role and Identity Management*. Finally, Figure 6 depicts a UML sequence diagram of the activities triggered by the INDIGO PaaS orchestrator upon the arrival of a user request. In the figure, the user is requesting the deployment of distributed and dockerized software components (Min.IO server/client, InfluxDB, Grafana) along the Edge-Cloud chain. The service orchestrator is in charge of deploying, configuring and pipelining the requested components. In the considered case, OpenStack IaaS is responsible for handling resources at the cloud end, while software deployment and configuration activities at the Edge node are supported by an Apache Mesos-based Docker image.

26. OpenStack: https://www.openstack.org/
27. OpenNebula: https://opennebula.io/
28. Mesos: http://mesos.apache.org/

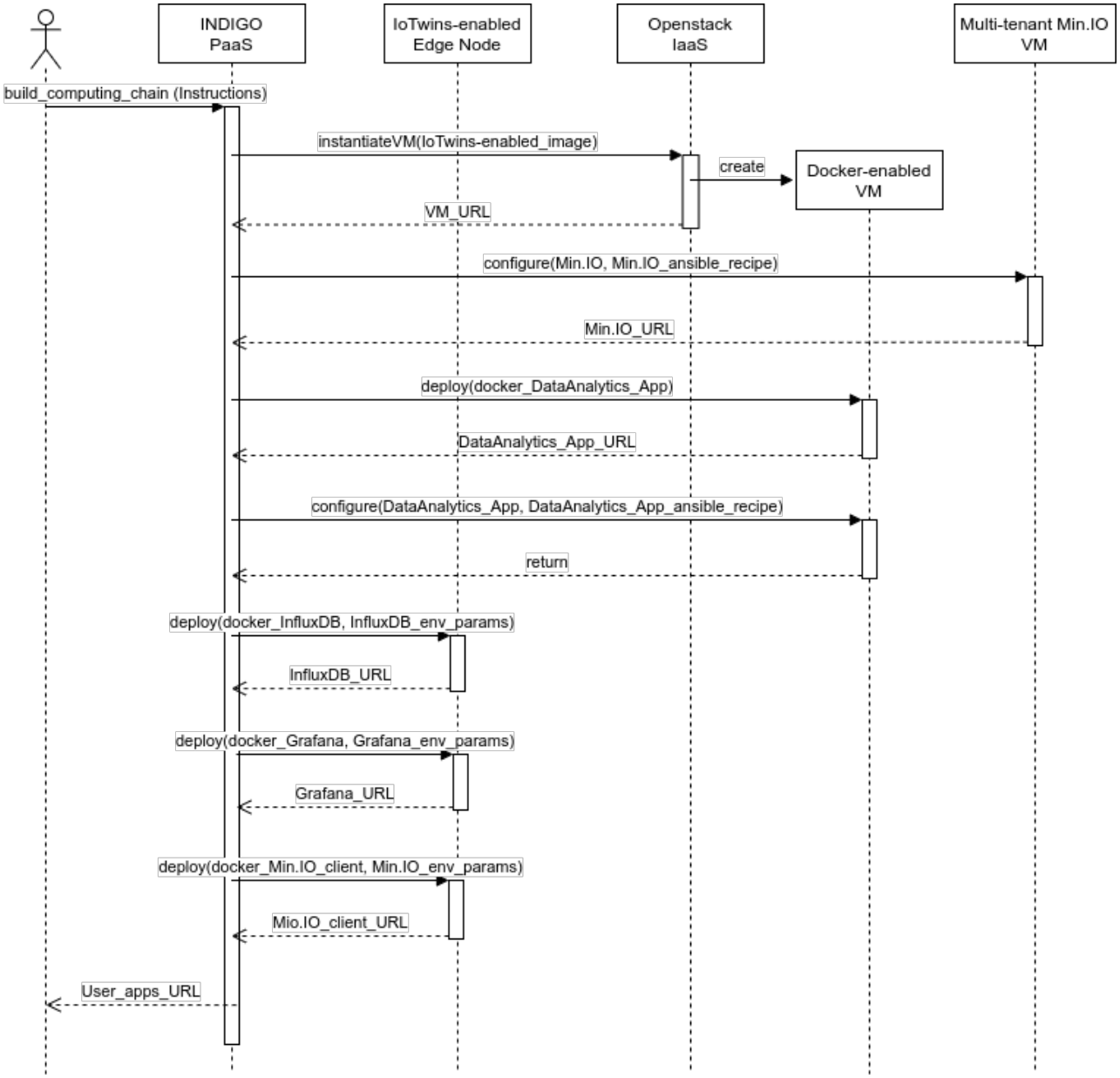

Figure 6: Application deployment: UML sequence diagram

4. Conclusion

In this paper we presented the requirements that a platform for the management of digital twins in industrial scenarios has to fulfill and the use-cases it has to enable and support. A reference architecture for platforms to manage such digital twins that was developed based on the derived requirements and use-cases was presented. Thereby the openness of the architecture allows an implementation using diverse technologies to exactly fit the application domains use-cases and technical requirements, which proved to be useful especially when an IoTwins platform is to be integrated with an existing system.

One possible implementation of the reference architecture was also presented. In the future, intensive experiments will be carried out on industrial partners' testbeds to make qualitative and quantitative assessment of the architecture in several domains (including manufacturing and energy management).

Acknowledgments

This work was partially supported by EU H2020 IoTwins Innovation Action project (g.a. 857191).

References

[1] Stephan Cejka, Felix Knorr, and Florian Kintzler. "Edge Device Security for Critical Cyber-Physical Systems". In: *2nd Workshop on Cyber-Physical Systems Security and Resilience (CPS-SR)*. 2019.

[2] Baotong Chen et al. "Edge Computing in Iot-Based Manufacturing". In: *IEEE Communications Magazine* 56.9 (2018), pp. 103–109.

[3] Wenbin Dai et al. "Industrial Edge Computing: Enabling Embedded Intelligence". In: *IEEE Industrial Electronics Magazine* 13.4 (2019), pp. 48–56.

[4] Igor Halenar et al. "Virtualization of Production Using Digital Twin Technology". In: *2019 20th International Carpathian Control Conference (ICCC)*. May 2019.

[5] Florian Jaensch et al. "Digital Twins of Manufacturing Systems as a Base for Machine Learning". In: *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. Nov. 2018.

[6] Abid Khan et al. "Towards Smart Manufacturing Using Spiral Digital Twin Framework and Twinchain". In: *IEEE Transactions on Industrial Informatics* (2020), pp. 1–1.

[7] Florian Kintzler et al. "Large Scale Rollout of Smart Grid Services". In: *2018 Global Internet of Things Summit*. 2018.

[8] Giuseppe Landolfi et al. "Design of a multi-sided platform supporting CPS deployment in the automation market". In: *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. May 2018.

[9] OASIS. *TOSCA Simple Profile in YAML Version 1.3*. http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3. Last accessed on 06-02-2021. Feb. 2020.

[10] Alexander Perzylo et al. "OPC UA NodeSet Ontologies as a Pillar of Representing Semantic Digital Twins of Manufacturing Resources". In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sept. 2019.

[11] Davy Preuveneers, Wouter Joosen, and Elisabeth Ilie-Zudor. "Robust Digital Twin Compositions for Industry 4.0 Smart Manufacturing Systems". In: *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*. Oct. 2018.

[12] Qinglin Qi and Fei Tao. "A Smart Manufacturing Service System Based on Edge Computing, Fog Computing, and Cloud Computing". In: *IEEE Access* 7.nil (2019), pp. 86769–86777.

[13] D. Salomoni et al. "INDIGO-DataCloud: a Platform to Facilitate Seamless Access to E-Infrastructures". In: *Journal of Grid Computing* 16.3 (Aug. 2018), pp. 381–408.

[14] Greyce N. Schroeder et al. "A Methodology for Digital Twin Modeling and Deployment for Industry 4.0". In: *Proceedings of the IEEE* (2020), pp. 1–12.

[15] Alexander Willner and Varun Gowtham. "Towards a Reference Architecture Model for Industrial Edge Computing". In: *IEEE Communications Standards Magazine* 4.4 (Aug. 2020), pp. 1–10. arXiv: 2008.04164.