This project has received funding from the European's Union Horizon 2020 research innovation programme under Grant Agreement No. 957258

Architecture for Scalable, Self-*, human-centric, Intelligent, Secure, and Tactile next generation IoT

ASSIST-IoT Technical Report #8

ASSIST-IoT: A Reference Architecture for Next Generation Internet of Things

Alejandro Fornés-Leal, Ignacio Lacalle, Carlos E. Palau, Paweł Szmeja, Maria Ganzha

Submitted to: EuCNC 2022, Grenoble, France (7-10 June 2022)

ASSIST-IoT: A Reference Architecture for Next Generation Internet of Things

Alejandro Fornés-Leal, Ignacio Lacalle, Carlos E. Palau
Communications Department
Universitat Politècnica de València
Valencia, Spain
{alforlea, iglaub}@upv.es,
cpalau@dcom.upv.es

Paweł Szmeja, Maria Ganzha
Systems Research Institute
Polish Academy of Science
Warsaw, Poland
{pawel.szmeja, Maria.Ganzha}@ibspan.waw.pl

***Abstract*—New requirements posed by the Next Generation IoT demand the design of novel reference architectures. Rooting on cloud-native concepts (such as micro-services-based systems and containerization), this paper proposes an architecture to cover all needed aspects. The paper elicits a wide description of components of the architecture (horizontal planes and vertical capabilities) while providing a formal definition of the views of the architecture, setting the ground for an upcoming deployment and validation in real scenarios in the short future. In particular, functional, node, deployment and data views are presented, each of them addressing the concerns of a group of particular stakeholders.**

Keywords—Next Generation IoT; Reference Architecture; Views; Distributed; Enablers; Data

I. INTRODUCTION

The Internet of Things (IoT) is evolving faster than ever. The unprecedented data explosion (both structured and -mostly- unstructured [1]) combined with the evolving capabilities of virtual infrastructures, set the scene for developing new scalable architectural paradigms.

Traditionally, IoT platforms have been focused on things' monitoring and actuation, being also vendor-locked and rather use-case specific. During the last few years, some initiatives have pushed forward a series of research (FIWARE, OM2M/OpenCDM, IoT-EPI, AIOTI) and commercial (AWS IoT, CISCO, IBM Watson) actions to devise advanced, modern, domain-agnostic (and, sometimes, even open-source) IoT architectures. However, there is still plenty of new requirement tides coming from different backgrounds (e.g., Big Data Analytics, AI, Cloud/Edge Computing, 5G connectivity, Industrial Data Spaces or new tactile interfaces like Augmented or Virtual Reality) aiming to be considered as explicit part of future IoT approaches. Those trends are paving the way to realize the IoT landscape into a portfolio of all-encompassing digital transformation enablers, supporting the Next Generation Internet (NGI) vision [2].

This work is part of ASSIST-IoT project, that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement 957258.

The objective of this paper is to propose a Reference Architecture (RA) for the so-called Next Generation IoT, covering the mentioned technological compass, and posing a deployment perspective to build on in years to follow. The designs exposed in this paper are strongly linked with the advances performed by H2020 ASSIST-IoT project, on track to be finalized by October 2023 and to be validated in three real-life scenarios on representative sectors (maritime ports, construction and automotive).

The paper is organized as follows: Section II presents a review of the RA terminology and paradigms utilized for their design, as well as a summary of existing solutions. Section III lays the technological foundation of the proposed architecture, detailing horizontal planes and vertical capabilities, as well as the concept and characteristics of the term "enabler". Section IV focuses on the rather formal description of the views of the architecture (Functional Data, Deployment and Node) while Section V reflects on future work and conclusions.

II. BACKGROUND

Reference Architectures (RAs) serve as blueprints to design, develop, implement and utilize IoT systems. They provide a set of guidelines, structures and models, which have to be flexible enough to be applicable in different domains and realized via different technological options, but without being so abstract that they become practically useless.

A. Concepts

Most of the concepts that are leveraged by RA designers have been defined by the ISO/IEC/IEEE 42010 standard [3]. The key ones are summarized here:

- *Stakeholder*: person, team, or entities that have a concern (also referred to as "interest") in a system. They can be technology (developers, administrators, maintainers, etc.) and non-technology related (users, acquirers, etc.).
- *Concern*: aspect of interest of stakeholders related to the RA. It includes needs, goals, requirements, quality attributes, dependencies, responsibilities, etc.

- *Views*: depict the architecture from the perspective of specific concerns, showing how these are addressed. The core ones are typically the Logical (or Functional), the Data (or Information), the Development, and the Deployment and Operational view [4].
- *Perspectives*: set of guidelines, strategies and tasks ensuring that a system considers a set of properties through many architecture views. In some RAs, these are referred to as system characteristics, however, in ASSIST-IoT the term "Vertical" is used, as it will also include specific functionalities to address specific cross-cutting concerns.

B. Architecture Paradigms

Nowadays, IoT architectures are growing in complexity: new features are being continuously included, workloads are extended and the number of components is rising. Architecture patterns are proper starting points to address such intricacy, supporting system components organization [5]. Some relevant architecture paradigms are the following (which in some cases can be combined together): layered, microkernel, event-driven, space-based, serverless and based on services. The latter, in turn, can be split into: (i) monolithic architecture, where its components form a unified, indivisible program or platform; (ii) service-oriented architecture (SOA), which decouples it into a set of modules, that are maintained independently and work together by means of an aggregation layer (messaging middleware); and (iii) microservices architecture, composed of lightly-coupled, independent services, which communicate with each other typically via API.

C. State of the Art

According to [6], RAs for NGIoT architectures must consider a set of features for providing or supporting: AI, Augmented Reality (AR), Digital Twins, Distributed Ledger Technologies (DLT), Edge Computing, Network Function Virtualization (NFV) and Tactile Internet. The first initiative bringing a RA for IoT was IoT-A [7]. It structures components modularly, in functional groups, namely: IoT process management, service organization, virtual entity, IoT communication, security and management. Other RAs have similar approaches, but with different modules/layers, like ITU-T Y.2060 ([8], in this case layered) and WSO2 ([9], layered). Others have evolved to include more advance features, mostly for addressing edge computing and virtualization, like RAMEC [10], OpenFog [11], ECC RA 2.0 [12], LSP 3D [13] or AIOTI HLA [14], being most of them multi-dimensional to split functionalities from properties and/or cross-cutting concerns. Finally, there are a set of RAs that include concerns related to industrial processes that are worth mentioning, like IIRA [15], RAMI 4.0 [16] and FAR-EDGE [17]. It should be mentioned that the degree of abstraction widely varies among them; whereas some do not go beyond recommendations, others specify implementation aspects in much higher details, even providing their own components for its realization. In any case, currently there is no single architecture addressing most of the features expected for NGIoT RAs, being that which is the aim of this paper.

III. Architecture Approach

A. Design principles

ASSIST-IoT proposes a RA governed by the following key principles and design decisions: (i) a **microservice** software architecture. Given the large number of technologies and features that can be used, keeping software in independent modules that can be later interconnected facilitates their maintenance and their use only when necessary; (ii) instantiation of these service in **containers**, hence decoupling them from the environment where they are executed; (iii) introduction of the abstract concept of ***enablers***, which bring a particular feature to the system; and (iv) **kubernetes** as the suggested underlying technology for orchestrating enablers, bringing a set of production-readiness advantages. Although this kind of impositions are not usually mentioned in RAs, it allows to provide useful guidelines and recommendations for actual realization, avoiding ambiguity and facilitating its use.

B. Conceptual architecture

ASSIST-IoT proposes a conceptual, multidimensional, decentralized approach, with layers (called *planes*) intersecting vertical blocks, as one can see in Fig. 1. This architecture follows the ISO/IEC/IEEE 42010 standard, and it is influenced by other RAs (mostly by AIOTI HLA, OpenFog and the LSP 3D architecture). The horizontal planes depict functionalities that can be grouped together, whereas *verticals* represent properties, cross-cutting concerns or NGIoT features that either needs cooperation among elements of more than one plane or that can exist in different planes, in an independent manner. Four planes are part of the architecture, namely:

1) Device and edge: comprises the physical elements that support an architecture realization, from servers and edge nodes to IoT devices, sensors, and network hardware.

2) Smart nework and control: composed of functions that facilitate virtualization and network connectivity, such as MANO, and virtualized network functions (e.g., virtual firewall, SD-WAN, VPNs, link aggregation, etc.).

3) Data management: groups functionalities related to data, from acquisition to sharing, fusion, aggregation, transformation, and storage.

4) Application and services: related to functions to be consumed by end-users, administrators and/or external systems. It makes use of functions from lower planes (and also verticals) to offer high-value applications for stakeholders.

Besides, five verticals have been identified:

1) Self-:* property of a system related to its autonomy or semi-autonomy, comprising specific operations that do not require human intervention (self-healing, self-configuration, self-awareness, self-organization, etc.).

2) Interoperability: property that ensures that, (i) at hardware level, equipment from different manufactures can communicate within a deployment, and (ii) at sofware level, services can share data thanks to the use of common formats or protocols, or the use of dedicated tools.

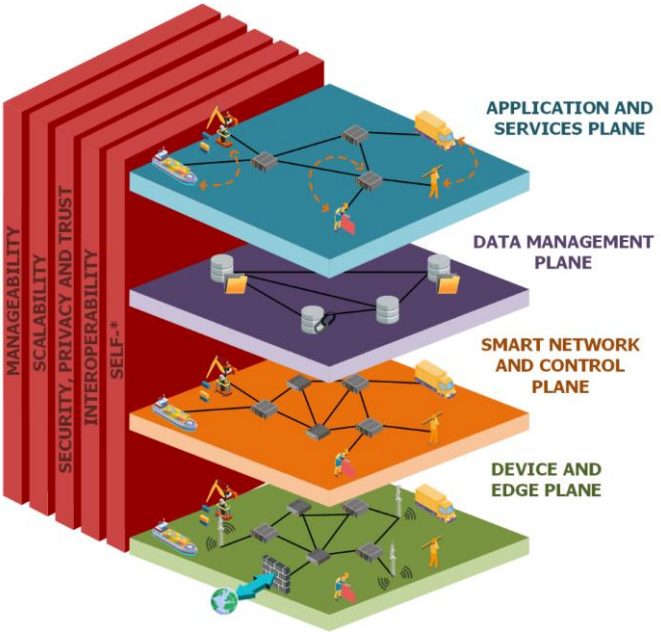

Fig. 1. ASSIST-IoT Conceptual Architecture

3) Security, privacy and trust: Set of properties of the architecture related to integrity and access restriction of data, as well as guarding against malicious threats, among others.

4) Scalability: trait to ensure proper system performance and dedication of resources in case of change of operational or business conditions or requirements. It comprises not just software, but also hardware and communication dimensions.

5) Manageability: related to the control of the lifecycle of the functions of other planes and verticals, from their instantiation and configuration to termination. It also comprises the management of devices and coordination of workflows.

In ASSIST-IoT, verticals in some cases are expected to be supported by specific functions, are not just inherent properties. For instance, self-* features include enablers with dedicated policies or AI/ML methods; cybersecurity requires dedicated monitoring agents; and data trust is to be ensured via distributed DLT logging mechanisms.

Similarly, horizontal planes and vertical properties are usually cross-dependent. To take an example, when forwarding sensors' data to a remote persistent system, some enablers in various planes must be involved (data management, network, application...) as well as the intervention of management traits for process configuration.

C. Enablers

In this RA, the term "enabler" is a conceptual abstraction that represents a collection of interconnected microservices (each of them referred to as "enabler component") that jointly provide a functionality for the system, and hence should fall under one of the specified planes or verticals. The rationale behind bringing this concept is mostly related to modularity. This entails that only those enablers required for addressing a specific scenario need to be considered.

Having functionalities realized as enablers has some advantages for development and maintenance. It allows having a conceptually more compacted view of the features available in the system, similarly to what happens in a SOA architecture, while having the benefits of working with microservices. Also, no language or communication interface is mandated for internal enabler designs, allowing flexible implementations.

Besides, enablers come with a set of common conventions that need to be followed. These conventions aim at facilitating enablers' integration in k8s environments:

- Encapsulation principle: enablers can communicate with each other only via an exposed interface (e.g., API, gRPC). Enabler components cannot be interacted directly from outside enabler's scope.
- Enablers should expose metrics and logs. ASSIST-IoT proposes following k8s *de facto* conventions. According to the first proposed design, metrics are exposed via a dedicated "/metrics" endpoint, following Prometheus-compatible format, whereas logs are sent out via *stderr* and *stdout* interfaces.

IV. Architecture Views

Representing all the information provided by a RA in a single, overloaded model would be impractical [4] and very difficult to follow by the stakeholders that aim at using it. The ASSIST-IoT RA follows a similar model as the 4+1 one from [18], considering four views: functional, node, deployment and data. For each of them, a summary of their scope and place in the realization of an architecture is provided.

A. Functional View

The goal of the functional view (also referred to as Logical view) is to define the functionalities provided by each horizontal plane, thus fulfilling the needs of the stakeholders and addressing their concerns. This view, whose functionalities are summarized in Fig. 2, are of interest for acquirers/users as well as developers and maintainers.

The device and edge plane is materialized primarily by processing elements and devices. Since the architecture is access-network agnostic, its realization will depend on the requirements of the business scenario (e.g., latencies, bandwidths, etc.). The nodes belonging to the edge must have enough processing power to support enablers from upper planes, as well as to include specific features or capabilities for the target environment (i.e., functionalities related to use cases, like localization awareness).

Following, for the smart network and control plane, eight enablers have been identified. The first and most important one is the smart orchestrator. It follows ETSI MANO principles [19] and current trends towards virtualization and Cloud-Native principles [20] (hence, towards containerization of network services rather than their instantiation in virtual machines). In the proposed architecture, role of the smart orchestrator does not stop at controlling the lifecycle of network-related functions but extends to the rest of enablers.

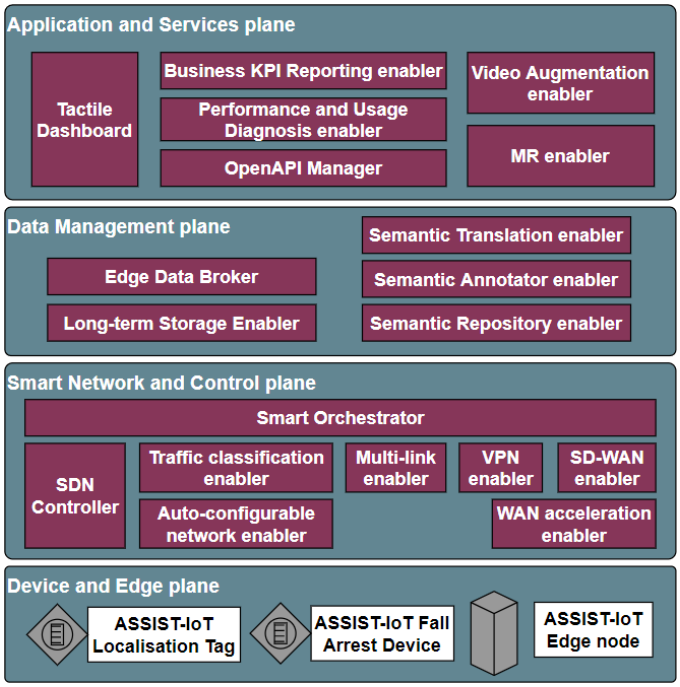

Fig. 2. Functional view representation – summary

Regarding SDN, a first trait can be found in the use of k8s itself, as cluster network can be controlled programmatically via dedicated plugins and controllers, managed by the orchestrator. Also, for those deployments with SDN-enabled equipment, three enablers (controller, auto-configuration and traffic classification) are envisioned. Apart from dedicated VNFs that may fall under its responsibility, the plane is completed with self-contained network realization, which is devoted to provisioning of private networks over public ones. This is realized via VPNs and SD-WAN technologies.

Services related to data governance and semantics are part of the data management plane. In short, it includes any service that distributes, stores or processes data. Semantic functions include a repository (hub of data models, schemas and ontologies), annotation (processing of data to be compliant with a particular semantic format), and translation (data transformation from one semantic format to another). So far, governance functions are delivered by two enablers: edge data broker, which aims at realizing pipelines for controlling the flow of the data over the system (i.e., distribute it within a path), and long-term data storage enabler, which will allow storing data in distributed fashion and be consumed by other enablers or applications.

The application and services plane encompasses any functions to be consumed by external systems or human users. It includes dashboards, for end users and administrators (tactile dashboard, including KPI reports, and performance and diagnosis indicators); human-centricity enablers to provide Augmented, Virtual or Mixed reality consumption of information managed by the architecture; and OpenAPI management to facilitate access to external users or systems.

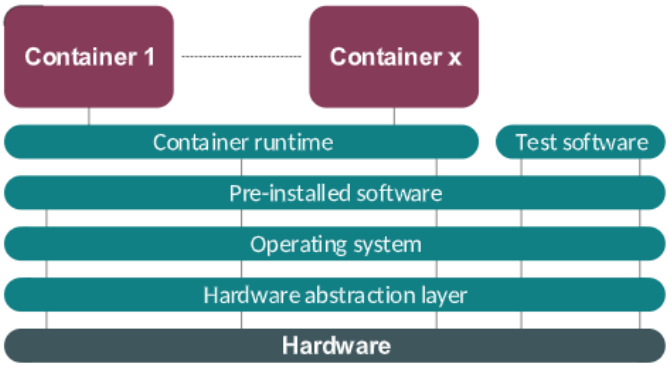

Fig. 3. Node view representation

B. *Node View*

This view maps to the logical element supporting the deployment of the functionalities depicted in the functional view. This view is not restricted solely to hardware equipment, but also to additional firmware and software elements as one can see in Fig. 3. This has an interest for both hardware developers and edge nodes or gateway providers.

As one of the principles of the architecture lies in the use of a k8s distribution (e.g., microk8s, k3s, k0s, etc.) as underlying container orchestrator, this must be part of pre-installed software, and the lower layers of the node view must support it (including the selected Operating System). In any case, the design of the nodes is quite open. Hardware (as well as the abstraction layer or firmware that allows its usage) depends on the requirements of the business scenario and its use cases. It stands not just for processing capabilities (e.g., RAM, CPU, GPU, etc.), but also for communication interfaces (e.g., WiFi, 5G, CAN, UWB, Ethernet, Serial...), and additional connectors (USB, for antennas, etc.). The same happens for pre-installed software (for instance, software needed to enable the use of GPU in the deployed containers), and regarding container runtime, any option compatible with k8s is allowed.

C. *Deployment View*

The deployment view responds to the necessity of exposing how an architecture is instantiated to address particular scenarios. This includes both the system topology and the deployment of enablers, including their combination to realize composite services. This view is of utility for developers, system administrators and maintainers.

The system topology of an architecture (see Fig. 4) can be composed of different interconnected tiers, each of them consisting of a set of physical elements, including networking and processing nodes. The following design principles drive the construction of this view for later implementations:

- The number of tiers comes from the business scenario and its use cases. Communication among nodes (from the same or different tiers) must be IP-based.
- A k8s distribution must be installed in all nodes. Clusters should be formed with nodes with similar capabilities, and logically each tier should have a k8s master and one (or more) k8s workers.

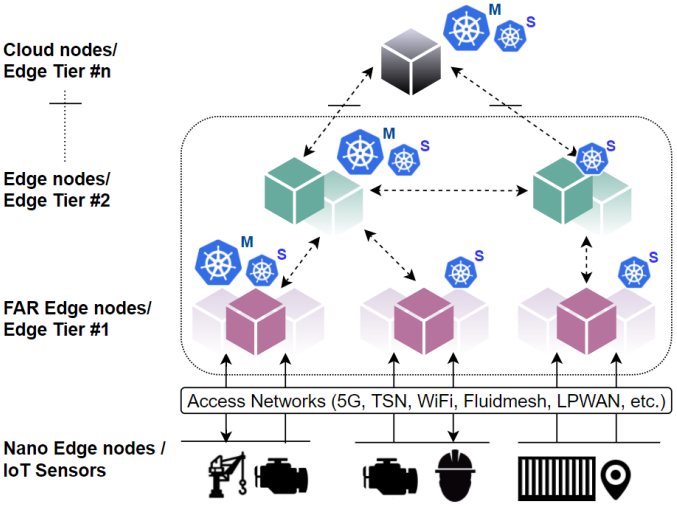

Fig. 4. Deployment view – topology representation

- A central node might be part of the deployment site or available at cloud premises. If nodes are part of different networks, connectivity must be secured via VPN or SD-WAN technologies.
- Decentralized, distributed use cases must be supported, without need of always interacting with nodes of upper tiers.
- In the scope of this view, a service is a combination of enablers, which may belong to different planes. The communication among (and towards) them can only take place via the exposed interface.

Some enablers are essential and should be part of any architecture realization to be considered an ASSIST-IoT deployment. These include: smart orchestrator, long-term data storage, edge data broker, tactile dashboard, OpenAPI manager, DLT logging and auditing enabler, basic security and manageability enablers.

D. Data View

Lastly, the data view aims at facilitating a high-level design of flow of data along the system (i.e., regarding their collection, transmission, processing and use), in an abstract, uncomplicated way, for addressing specific use cases. This view can be of particular interest for data engineers as well as for developers and maintainers.

As an architecture that supports decentralized use cases, specific flows of data are not imposed. To construct them, the concept of "data pipelines" is introduced. Pipelines are graphical descriptions with textual elements that show how data are managed by the system, from its generation/capture (source) through their "path" (processed by enablers) towards their destination (sink), where they are consumed. At sources, data is represented as a "message", which should include relevant information (entities, measurements, events, etc., without requiring the degree of formality of typical UML diagrams). Paths between enablers are then indicated, and the processing carried out in each of them depends entirely on their respective functionalities. Sinks can be user applications, other

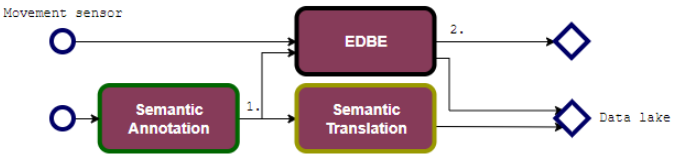

Fig. 5. Example of data pipeline

enablers (e.g., long-term storage) or even other data pipelines. An example of data pipeline is shown in Fig. 5.

V. Conclusions and Future Work

This paper presents a novel reference architecture for the Internet of Things, considering new requirements and recommendations posed by the Next Generation IoT. This architecture is driven following cloud-native patterns, adapted to the edge. At a conceptual level, it is organized in planes, which depict functionalities that can be logically grouped together, and verticals, which include properties, cross-cutting concerns and NGIoT features that can exist in multiple planes or require cooperation among them.

The constant technological evolution claims for a blueprint architecture that takes it into account, being flexible enough to avoid fast obsolescence. The modularity brought by the conceptual abstraction of enablers plays this role, facilitating the inclusion, modification or deletion of plane and vertical features over time. Besides, being inspired in former works and standards, the architecture is represented in four separated views rather than in an overloaded, all-encompassing model, namely functional, node, deployment and data view. Each of them is intended to provide relevant information targeting specific stakeholders, offering guidelines and recommendations for an actual realization, while trying to avoid too much abstraction or ambiguity.

As a yet novel architecture, there is still room for enhancing and extending the information provided by the views in order to ease its application for realizing specific use cases, or even constructing additional views for addressing other stakeholders' concerns. To this end, the project is developing its own set of enablers to facilitate its application and evaluation in different scenarios.

Being part of an on-going research action, the proposed architecture will be actually tested in real, relevant use cases. In particular, it will be used to: (i) solve latency issues while carrying virtual reality applications in a maritime terminal, (ii) allow an accurate fall arrest detection of smarter (wearable-wearing) workers in construction plants, and (iii) improve the training and real-time inspection of defects in car surfaces, among others.

References

[1] J. ho Park, M. M. Salim, J. H. Jo, J. C. S. Sicato, S. Rathore, and J. H. Park, "CIoT-Net: a scalable cognitive IoT based smart city network architecture," Human-centric Comput. Inf. Sci., vol. 9, no. 1, pp. 1–20, Dec. 2019.

[2] NGI Project, "A Vision of the Future Internet," 2020.

[3] ISO/IEC/IEEE 42010, "Systems and software engineering - Architecture description," 2011.

[4] N. Rozanski and E. Woods, Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison Wesley, 2011.

[5] A. Sharma, M. Kumar, and S. Agarwal, "A Complete Survey on Software Architectural Styles and Patterns," in Procedia Computer Science, 2015, vol. 70, pp. 16–28.

[6] NGIoT Project, "D3.1. IoT research, innovation and deployment priorities in the EU," 2020.

[7] M. Bauer et al., "Final architectural reference model for the IoT," 2013.

[8] ITU-T, "Y.2060: Overview of the Internet of things," 2012.

[9] WSO2, "A Reference Architecture for the Internet of Things," 2015.

[10] A. Willner and V. Gowtham, "Towards a Reference Architecture Model for Industrial Edge Computing," Comput. Sci., 2020.

[11] OpenFog Consortium, "OpenFog Reference Architecture for Fog Computing," 2017.

[12] ECC and AII, "Edge Computing Reference Architecture 2.0," 2017.

[13] CREATE-IoT Project, "D6.3. Assessment of convergence and interoperability in LSP platforms," 2020.

[14] AIOTI WG Standardisation, "High Level Architecture (HLA) Release 5.0," 2020.

[15] IIC, "The Industrial Internet of Things Volume G1: Reference Architecture v1.9," 2019.

[16] VDI/VDE Society Measurement and Automatic Control (GMA), "Status Report Reference Architecture Model Industrie 4.0 (RAMI4.0)," 2015.

[17] FAR-EDGE Project, "D2.4. FAR-EDGE Architecture and Components Specification," 2017.

[18] P. B. Kruchten, "The 4+1 View Model of Architecture," IEEE Softw., vol. 12, no. 6, pp. 42–50, 1995.

[19] ETSI, "GS NFV-MAN 001 Network Functions Virtualisation (NFV); Management and Orchestration," 2014.

[20] 5G-PPP Software Network Working Group, "Cloud-Native and Verticals' services," 2019.