



D1.2

IoT meta-architecture, components, and benchmarking

WORKPACKAGE	WP1	PROGRAMME IDENTIFIER	H2020-ICT-2020-1
DOCUMENT	D1.2	GRANT AGREEMENT ID	957246
REVISION	V2.0	START DATE OF THE PROJECT	01/10/2020
DELIVERY DATE	30/09/2021	DURATION	3 YEARS
(revised version: 01/08/2022)			

© Copyright by the IoT-NGIN Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246



DISCLAIMER

This document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of its content.

This document may contain material, which is the copyright of certain IoT-NGIN consortium parties and may not be reproduced or copied without permission. All IoT-NGIN consortium parties have agreed to full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the IoT-NGIN consortium as a whole, nor a certain party of the IoT-NGIN consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk and does not accept any liability for loss or damage suffered using this information.

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Bosch Sistemas de Frenado S.L.U. (BOSCH), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), Optimum Technologies Piroforikis S.A. (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelxis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP1
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	30/09/2021
ACTUAL DATE OF DELIVERY	30/09/2021 (revised version: 01/08/2022)
DOCUMENT TITLE	IoT meta-architecture, components, and benchmarking
AUTHOR(S)	AALTO, EBOS, EDD, EMOT, RWTH
REVIEWER(S)	RWTH, SYN
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	IoT, Meta-architecture, architecture, pattern, view, design

Document History

Version	Date	Contributor(s)	Description
V0.1	10/7/2021	AALTO	First draft, Table of Contents
V0.2	16/7/2021	AALTO	Background and analysis perspectives
V0.3	20/7/2021	AALTO	Meta-architecture overview
V0.4	24/7/2021	AALTO	Architectural patterns conclusions, quality vertical section. Updated meta-architecture visualization.
V0.5	30/7/2021	AALTO	Updated commercial and EU research analysis and data models
V0.6	9/9/2021	AALTO, EBOS, EDD,	Integrated inputs from various parties
V0.7	10/9/2021	AALTO, EBOS, EDD, SYN	Integrated inputs from various parties, document consolidation
V0.8	23/9/2021	AALTO, CMC, EBOS, EDD, EMOT, RWTH, SYN	Integrated partner input. Added components section and updated architectural patterns.
V0.9	26/9/2021	AALTO, ABB, ATOS, CMC, EBOS, EDD, EMOT, RWTH, SYN	Submitted for peer review
V0.9.1	29/09/2021	RWTH	Peer review comments
V0.9.2	29/09/2021	SYN	Peer review comments
V0.9.3	29/09/2021	AALTO, ABB, ATOS, CMC, EBOS, EDD, EMOT, RWTH, SYN	Consolidated document
V0.9.4	30/09/2021	SYN	Quality check
V1.0	30/09/2021	AALTO	Final version
V2.0	29/07/2022	SYN	Updates on the existing IoT platforms presentation, to be more coherent.

Table of Contents

Document History	4
Table of Contents	5
List of Figures	7
List of Tables	8
List of Acronyms and Abbreviations.....	10
Executive Summary	12
1 Introduction.....	13
1.1 Intended audience.....	14
1.2 Relation to other activities.....	14
1.3 Document Overview.....	15
2 Existing IoT platforms.....	17
2.1 Open-source perspective	18
2.2 Commercial perspective	22
2.3 EU research perspective.....	27
3 IoT architectural patterns overview and analysis	32
3.1 Layered architecture	32
3.2 Microservices-oriented architecture	32
3.3 Event-driven architecture.....	34
3.4 Lambda architecture	35
3.5 Software-defined networking (SDN)	36
3.6 Edge computing	37
3.7 Wireless sensor networks architectures	37
3.8 5G network exposure	38
3.9 5G network slicing	39
3.10 Gateways	40
3.11 Distributed ledger technologies (DLT)	40
3.12 Serverless computing	41
3.13 Self-sovereign identities (SSI)	42
3.14 Data federation.....	42
3.15 Digital twins	43
3.16 Reference models.....	44

4	The IoT-NGIN meta-architecture	45
4.1	Definition.....	45
4.2	Key meta-architecture viewpoints.....	45
4.3	Element view.....	48
4.4	Architectural Patterns Vertical.....	58
4.5	Quality Vertical	58
4.6	ML and AI layer.....	61
4.7	Compliance to the IoT-NGIN meta-architecture	63
5	The IoT-NGIN architecture	65
5.1	Mapping the IoT-NGIN architecture to the meta-architecture	69
6	Common IoT data models	72
6.1	International Data Spaces	72
6.2	W3C Web of Things (WoT)	73
6.3	FIWARE and ETSI Context Information Management: NGSI-v2 and NGSI-LD	74
7	Benchmarking and test reports	77
7.1	Methodology	77
7.2	Implementation of methodology for benchmarking	79
8	Conclusions	82
9	References	83

List of Figures

Figure 1: IoT-NGIN PERT chart.....	15
Figure 2: IoT Layers [38].....	33
Figure 3: Microservices Architecture.....	34
Figure 4: Event Broker components.	35
Figure 5: Lambda architecture elements.....	36
Figure 6: Wireless Sensor Networks overview.	38
Figure 7: 5G network slicing structure.	40
Figure 8: Serverless computing architecture.	42
Figure 9: Digital Twins Architecture [49].	43
Figure 10: Early model of IoT ARM and its functional view [51].....	44
Figure 11: IoT-NGIN Meta-architecture.....	47
Figure 12: MLaaS Architecture, first version	62
Figure 13: The high-level architecture of IOT-NGIN.....	65
Figure 14: IDS Information Model. Adapted from IDS	73
Figure 15: WoT Building Blocks	74
Figure 16: NGSI-LD and federated architecture. Adapted from NGSI-LD.....	76
Figure 17: Agile methodology for product evaluation.....	77

List of Tables

Table 1: Common architectural views.....	13
Table 2: IoT approaches and platforms selected for further study	18
Table 3: Open source platforms.....	18
Table 4: OpenRemote Specifications	20
Table 5: Mainflux Specifications	20
Table 6: Thingsboard Specifications.....	21
Table 7: Kaa Specifications.....	21
Table 8: DeviceHive Specifications	22
Table 9: Commercial IoT Platforms.....	22
Table 10: Azure IoT Hub Specifications	24
Table 11: AWS IoT Core Specifications.....	25
Table 12: Google Cloud IoT Specifications	25
Table 13: PTC ThingWorx Specifications.....	26
Table 14: Particle Specifications.....	26
Table 15: IBM IoT Specifications.....	27
Table 16: Cumulocity IoT Specifications	27
Table 17: European Research Projects relevant to IoT-NGIN.....	28
Table 18: IoT-NGIN relevant research projects and their relationship with IoT-NGIN.	29
Table 19: Elements of the Things functional group.	48
Table 20: Elements of the Fog-Edge functional group.....	49
Table 21: Elements of the Analytics functional group.	50
Table 22: Elements of the automation functional group.	51
Table 23: Elements of the Infrastructure functional group	51
Table 24: Elements of the Infrastructure/Cloud subgroup.....	52
Table 25: Elements of the Infrastructure/Container as a Service subgroup.	53
Table 26: Elements of the Infrastructure/5G networking subgroup.....	53
Table 27: Elements of the Federation functional group	54
Table 28: Elements of the Workloads functional group.....	55
Table 29: Elements of the Workloads/Services subgroup.....	56
Table 30: Elements of the Workloads/Middleware subgroup	57
Table 31: Elements of the Workloads/Data subgroup.....	57
Table 32: IoT-NGIN meta-architecture compliance matrix.....	63

Table 33: The meta-architecture functional groups	64
Table 34: Compliance of the IoT-NGIN architecture with the meta-architecture; architectural patterns.	67
Table 35: Compliance of the IoT-NGIN architecture with the meta-architecture; Quality Vertical.	68
Table 36: Compliance of the IoT-NGIN architecture with the meta-architecture; Elements View.	70
Table 37: Sample QoS metrics.	78
Table 38: UCs of IoT-NGIN.	79
Table 39: KPIs for UC1.	80
Table 40: Example of test report format for QoE.....	81

List of Acronyms and Abbreviations

AAS	Asset Administration Shell
API	Application Programming Interface
AMQP	Advanced Message Queuing Protocol
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
DDS	Data distribution service
DLT	Distributed Ledger Technology
DTLS	Datagram Transport Layer Security
GE	Generic Enabler
EA	Enterprise Architecture
gRPC	open-source Remote Procedure Call
IaaS	Infrastructure-as-a-service
IAM	Identity and Access Management
IoT	Internet of Things
JWT	JSON Web Token
KPI	Key Performance Indicator
LL	Living Lab
LSP	IoT Large Scale Pilot
LWM2M	Lightweight Machine to Machine
M2M	Machine-to-machine
MCM	Machine-cloud-machine
MQTT	Message Queuing Telemetry Transport
MQTT-SN	MQTT For Sensor Networks
mTLS	Mutual TLS Authentication
NFV	Network function virtualisation
NGSI	Next-Generation Services Interface
NGSI-LD	NGSI Linked Data
OPC-UA	Open Platform Communications United Architecture
PaaS	Platform-as-a-service
SaaS	Software-as-a-service
SDK	Software Development Kit
SDN	Software-defined network

SMQT	Secure MQTT
SSI	Self-sovereign identity
TCP	Transmission Control Protocol
TD	Thing Description
TLS	Transport Layer Security
UAT	User Acceptance Tests
UDP	User Diagram Protocol
UDT	UDP-based Data Transfer Protocol
W3C	The World Wide Web Consortium
WoT	Web of Things

Executive Summary

After years of research, hype and steady growth, the Internet of Things (IoT) has lived up to expectations and has entered into mainstream business use. The anticipated growth in the next years is high, with Fortune Business Insights™ in its report, titled "Internet of Things (IoT) Market, 2020-2027" [1] projecting the global IoT market size to reach USD 1463.19 billion by 2027. The same report suggests that significant push in the IoT market is expected by the increasing demand for artificial intelligence and twin technology and precision farming worldwide. Europe is leading the IoT market with IoT spending in Europe estimated to reach USD 202 billion in 2021 and continue to experience double-digit growth through 2025, according to International Data Corporation's (IDC) Worldwide Semiannual Internet of Things Spending Guide [2].

The Next Generation IoT as part of Next Generation Internet (IoT-NGIN) project introduces novel research and innovation concepts, acting as the "IoT Engine" which will fuel the Next Generation of IoT as a part of the European Next Generation Internet. A key project objective is to uncover a patterns based meta-architecture that encompasses evolving, legacy, and future IoT architectures.

This document constitutes Deliverable "D1.2: IoT meta-architecture, components and benchmarking", which is an output of Work Package (WP) 1, entitled "Next Generation IoT Requirements & Meta-Architecture. The main achievements towards the definition of the IoT-NGIN meta-architecture analyzed in the present document include:

- Identification and analysis of the state-of-the-art IoT platforms and frameworks, covering open-source, commercial and EU research level perspectives.
- Identification and analysis of 16 distinct IoT architectures.
- An initial concept and technical specifications of the IoT-NGIN meta-architecture has been defined around four key artifacts, namely IoT architectural pattern vertical, domain horizontal, quality vertical, and element view.
- Detailed analysis of the individual components within the meta-architecture has been conducted and compliance to the IoT-NGIN meta-architecture has been discussed.
- The IoT-NGIN architecture, complying to the meta-architecture, has been presented, identifying fine-grained components across functional blocks for both the IoT and Edge/Cloud nodes.
- Common IoT data models and standardization approaches have been identified to ensure platform's interoperability and adaptability.
- Initial test reports format and benchmarking methodology have been defined based on Quality of Service to assess the performance of the system through already defined KPIs, and on Quality of Experience to quantify the user experience.

Following the current IoT technological trends, as well as key project findings as they become available, the IoT-NGIN meta-architecture is being continuously updated. The next version of the IoT-NGIN meta-architecture, featuring such updates, is expected to be released in the last quarter of 2022.

1 Introduction

An architecture of a system is the structured set required to proficiently reason about the system. The structures consist of elements, relations, interactions and properties of the system under study. Traditionally in software engineering, architecture refers to software systems and its elements. Architecture plays a vital role in complex system development and in the system's capability of inter system communication or of changing its hierarchy.

The meta-architecture takes a higher-level view and collects together architecturally significant choices, patterns, components, viewpoints and quality attributes that need to be considered when designing and implementing individual systems. The meta-architecture provides a foundation for the architecture strategy and system design.

The aim of the meta-architecture work in IoT-NGIN is to provide a research-informed framework for designing and implementing IoT solutions in different usage scenarios. The meta-architecture collects key quality requirements, architectural patterns and high-level system components. The meta-architecture provides an overall framework for individual system implementations. The motivation is to enable reuse of existing IoT technologies and solutions to new domains and assist structured, informed IoT platform design. Consequently, the meta-architecture is designed to be extensible to make room for new technologies and computing paradigms and allow the effective and seamless exploitation and exploration of both historical and real-time IoT data.

The IoT-NGIN meta-architecture is based on analysis of current IoT platforms and standards. The analysis of current approaches covered key open-source and commercial IoT platforms and was further extended by EU research perspective covering most relevant EU research projects in recent years. The architectural patterns included into the meta-architecture are based on expert advice and knowledge identified in previous open-source, commercial and EU research projects complemented by the research conducted within IoT-NGIN.

An architectural view represents the design of software system from a specific point of view. The architectural views commonly used in software design are listed in Table 1. Viewpoints, on the other hand, include reusable information, design patterns, templates, and conventions to build and read a specific view. Viewpoints are valuable because one can reuse architecturally valuable information in designs. This approach allows for better management of complexity because viewpoints can be used to instantiate the views. Following the ideas of views and viewpoints, the developed meta-architecture aims to build and visualise a specific strategic viewpoint for IoT-specific applications.

Table 1: Common architectural views.

View	Description
Context view	Environment and stakeholder dynamics.
Functional view	Functional architecture and critical component/elements.
Development view	Source code, class structure, dependencies.
Information view	Information flow and static information.
Item Deployment view	Infrastructure and physical distribution.

Item Operational view	Operation, management, support of a system in production environment.
Item Concurrency view	Processes and threads.

1.1 Intended audience

The document could be especially useful to IoT stakeholders interested in adopting the IoT-NGIN meta-architecture. IoT and edge hardware manufacturers, IoT solution providers, but also 5G and AI-related stakeholders could get insights on the architectural patterns for their fields of interest. Moreover, the document provides technical specifications, analyzes architectural viewpoints and data models and introduces methodology for benchmarking and testing, which could help IoT solution providers, technology providers and developers enhance their IoT solutions, developments or products, adapting them into or extending the IoT-NGIN meta-architecture.

Policy makers and regulation bodies are also among the interested stakeholders, since the IoT-NGIN meta-architecture could be exploited for cross-domain applications at municipal, national or regional level.

Finally, the report is useful internally, to the members of the development and integration team of the IoT-NGIN consortium, along with the Open Call winners, but also to the whole Consortium for validation and exploitation purposes. Useful feedback could be also received from the Advisory Board, including both technical and impact creation comments.

1.2 Relation to other activities

Since this document collects the requirements and ratifies the technological and design perspectives of the IoT-NGIN meta-architecture and the IoT-NGIN project architecture, it is relevant to the majority of the technical project activities.

Indeed, with reference to Figure 1, depicting the project PERT chart, the work of WP1 (and the meta-architecture in particular) directly affects the definition of activities in the technical WPs of the project, implicitly hinting over design and technology implementation decisions, offering guidelines regarding the network design (WP2), requirements regarding the global use of Artificial Intelligence (AI) and Machine Learning (ML) in WP3, the introduction of Augmented Reality (AR) and tactility to the IoT landscape (WP4), as well as the cybersecurity and data privacy aspects ruling the IoT-NGIN operations (WP5). At the same time, the requirements for the meta-architecture stem from the actual needs of the Living Labs and their associated Use Cases (WP7) as well as from the overall technology integration work of the project (WP6), particularly when it comes to addressing the needs of the manageability perspective of the IoT-NGIN activities and deployments, and vice-versa.

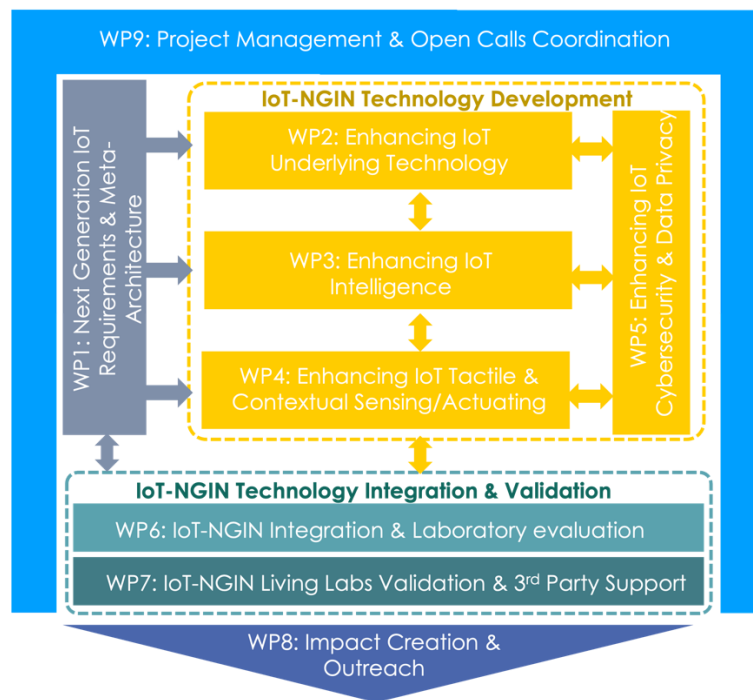


Figure 1: IoT-NGIN PERT chart.

Being a WP1 deliverable, this document is also relevant to the internal WP1 activities. Task 1.1 posed several functional and non-functional requirements to be covered by the meta-architecture definition, whereas the Task1.2 activities formulated most of the deliverable content. The verification framework was developed in the context of Task 1.3. Finally, Task 1.4 identified the relevant existing technologies, frameworks, products and projects that are relevant to the scope of the meta-architecture definition and design. This document should be considered as a living document, its contents being affected by the inherent advancements of technology per se (identified in the context of Task 1.4) and of the IoT-NGIN Living Labs (LLs) requirements evolution.

1.3 Document Overview

The present deliverable is divided into seven chapters, as follows.

Chapter 1 introduces the **motivation and objectives** of the deliverable. Moreover, it explains **the inter-relationships of this deliverable with other Work Packages** of the project since the defined meta-architecture will form the basis that the rest of the project will be built upon.

Chapter 2 provides an **overview of several IoT technologies, platforms and solutions** covering all three perspectives, commercial, open-source and EU projects and it further presents technological specifications of each platform.

Chapter 3 describes **architectural patterns** observed in the analysed platforms in detail showing strengths and weaknesses of each one and suitability for various applications.

Chapters 4 is the main output of this deliverable. It explains the concept of meta-architecture and gives a detailed explanation of the **derived IoT-NGIN meta-architecture** based on the architectural pattern analysis described in chapters 2 and 3. This is the **initial concept of the meta-architecture** that will be used in the implementation of the IoT-NGIN use cases. It also

presents the **element view** of the meta-architecture, which provides a general technological view with a focus on its **functional groups and individual components**. Moreover, it presents the non-functional **architectural patterns vertical** which discusses **common quality attributes included** in the architecture. It also discusses the overarching **machine learning and artificial intelligence** layer that is developed as a part of IoT-NGIN Machine Learning as a Service platform. This chapter posits a preliminary set of criteria for IoT-NGIN meta-architecture compliance. As IoT-NGIN uses existing and mature networks, **compliance with several standards was the main perspective of the discussion** in order to ensure interoperability and adaptability of the IoT-NGIN platform.

Chapter 5 presents the modular **IoT-NGIN architecture**, describing its components in every functional group. The chapter also discusses the **compliance of the IoT-NGIN architecture to the meta-architecture** defined in Chapter 4.

Chapter 6 describes various **data models** implementation options that are potential candidates being **compatible with the IoT-NGIN requirements** whilst they are relevant to data sovereignty and federated communications.

Chapter 7 provides the initial perspective towards **benchmarking** of the IoT-NGIN results by defining **test reports formats** and benchmarking for the **validation of the Key Performance Indicators (KPIs)**. The chapter provides the methodology for the verification of the results and focuses on two metrics, the **Quality of Service**, which assesses the performance of the platform through the KPIs and the **Quality of Experience**, which quantifies the user experience.

Chapter 8 concludes with the **main findings and outputs of the report** as well as an initial perspective on the **next steps** towards the implementation of the meta-architecture in the IoT-NGIN platform.

2 Existing IoT platforms

To assess current state of the art in IoT architectures, this section analyses various identified IoT technologies, solutions and approaches. The IoT-NGIN meta-architecture is constructed based on these identified IoT platforms and standards. The architectural patterns included into the meta-architecture are based on expert advice and knowledge identified in previous open-source, commercial and EU research projects complemented by information provided by IoT-NGIN project partners.

Most of the analysed entities in open-source and commercial perspectives can be labelled as cloud-based IoT platforms, which complement their offering with on-premise and hybrid services. Cloud IoT platforms offer versatile connectivity and device management, while simultaneously providing capabilities for data ingestion and processing. Many cloud platforms form bi-directional communications, enabling communication between the devices and the cloud back-end services. IoT applications can be implemented with a diverse set of heterogeneous IoT devices, which push sensor and event data to the cloud through a network. In this task, the analysis of IoT solutions is focused on the following characteristics [3]:

- **Data communication and connectivity** are requirements for remote orchestration of devices and transferring of data. There is a profound need to have secure and high-quality communication between the devices, applications and the cloud infrastructure. The communication types are often referred to as different abbreviations: Device to Cloud (D2C), Cloud to Device (C2D), Device to Device (D2D) also frequently referred to (M2M), Device to Application (D2A) and Device to Gateway (D2G). The set of communication types enhances the notion of complex communication in IoT. Data can be collected, transmitted and stored in various layers and parts of the system. To answer the challenge, cloud IoT platforms often contain a message broker for sending and receiving events and messages from devices and gateways.
- **Rules and business processing** is mandatory in the event-based IoT environments. The logical flow of data and information depend on the application and use case. A rule engine or similar function is often embedded in the system to integrate the technical flows with business logic and create useful action triggers.
- **Integration** with other platforms, devices, services and development tools unleash the power of cloud IoT platforms and are the key reasons for their dominance. In this context, integration is discussed as the general connector of the high-level components. Many of the analysed technologies offer development performance and various business interfaces through SDKs and APIs.
- **Potential solution-specific features** are analysed on a case-by-case basis. No technology is identical and as a sub-objective of this analysis is not only to identify the recurrent architectural approaches but also discover valuable solutions that can be applied to the IoT-NGIN project. While the developed meta-architecture does not address the specific technologies, the discoveries will be useful in other parts of the project.
- **Security** is one of the most important characteristics across other analysis categories. Security is one of the cornerstones of IoT-NGIN, thus it is considered as a standalone characteristic. The security features are highlighted especially in the device level, data and communications and infrastructure security. In addition, some special cryptography features with industry-wide adoption are included in the analysis.

- As in any business, **cost** is an important factor, but it is delimited out of the scope of this analysis. It is still a crucial notion that design, building and operating costs differ largely depending on the application, scale and the whole operation environment, such as service providers and licensing.

Based on a review of state-of-the-art IoT platforms and other prior works, a number of platforms presented in Table 2 were selected for further influencing the design of the IoT-NGIN meta-architecture.

Table 2 shows an overview of the existing IoT approaches and platforms.

Table 2: IoT approaches and platforms selected for further study

Type	Existing relevant projects
Open source	FIWARE [4], W3C Web of Things (WoT) [5], DeviceHive [6], ThingsBoard [7], Kaa [8], OpenRemote [9], Mainflux [10].
Commercial	Azure IoT [11], AWS IoT [12], Google Cloud IoT Core [13], PTC Thingworkx [14], IBM Watson IoT [15], Cumulocity IoT [16], Particle [17].
EU Research	The IoT European Large-Scale Pilots (LSP) Programme [18] (CREATE-IoT, SynchroniCity, MONICA, AUTOPILOT, ACTIVAGE), GAIA-X, INTER-IoT, AGILE, other related projects identified by IoT-NGIN (SOFIE, NRG-5, OPEN DEI, SOGNO, AI4EU, MATILDA, 5GCity, PRIMO 5G)

2.1 Open-source perspective

There are various drivers for open-source development, but the most significant are velocity of innovation, focus on interoperability and vendor lock-in avoidance, and easier experimenting. Open-source technologies often combine other open standards and robust open-source implementation.

Table 3 below shows the analysed open source architectures and their general description.

Table 3: Open source platforms.

Name	Description
FIWARE	FIWARE is an open-source framework and platform that features to build its components for smart solutions across the domains of Smart AgriFood, Smart Cities, Smart Energy, and Smart Industry. FIWARE provides IoT capabilities through cloud-based context information management and big data services and is built around a FIWARE Context Broker. FIWARE Generic Enablers (GEs) enable rapid development of applications and services. GEs are often interfaced with REST APIs, which allow third-party components to build complex solutions, such as real-time data analysis and processing, predictive maintenance, and language interpreters [4].

W3C Web of Things (WoT)	WoT focuses on harmonizing the IoT landscape by developing and extending existing and already standardised technologies. WoT is based on building blocks that describe IoT devices and services, referred to as the Things, at various levels.
DeviceHive	DeviceHive is an open-source IoT cloud service management platform, licensed under the Apache License Version 2.0. It consists of the communication layer, control software and multi-platform libraries and clients. DeviceHive positions itself as a platform with versatile deployment options in public, private and hybrid clouds [6].
ThingsBoard	ThingsBoard is an open-source IoT platform for specializing in data processing, and device orchestration. ThingsBoard is designed to be scalable, robust and efficient. Yet, it minimises trade-offs in fault-tolerance, durability and customization options. It offers connectivity via standard IoT protocols and supports cloud and on-premises deployments [7].
Kaa	Kaa is an end-to-end IoT platform applicable to IoT projects of any scale. It provides a wide offering of features for developers to build applications such as smart products, flexible cloud-based device management, or end-to-end data processing. Kaa Cloud operates on the Platform-as-a-Service (PaaS) model. Kaa operates through distinct endpoints that feed to Kaa protocol communication. Depending on the application, the transferred information can be for example command execution, data collection operation, or configuration instructions [8].
OpenRemote	OpenRemote is an open-source IoT platform that integrates a variety of devices, assets and other data sources into a single asset and data management solution. It allows designing applications and workflows specific to customer problems. Visualizing and analyzing the managed data is convenient and OpenRemote offers tools for reporting and progress measurement and has easily accessible web user interfaces and consoles [9].
Mainflux	Mainflux is a microservice-based, high-performance and secure open-source IoT platform with end-to-end development capacity of IoT solutions, applications and smart products. With Apache 2.0 license, Mainflux offers transparency, control, and support community for testing, bug fixes and more [10].

Table 4 through to Table 8 below discuss in more detail, the characteristics and common features of the platforms presented in Table 3.

Table 4: OpenRemote Specifications

Feature	Detail
General architecture	Agent-based architecture
Data communication and connectivity	Protocol Support Agent. TCP/HTTP, MQTT, REST API Manager, WebSocket endpoint. Edge Gateways, KNX, ArtNET/DMX, Z-wave
Rules and business processing	Rules Engine for automation: JSON and Flow Rules Object Model. Prediction and Optimisation Models. Messaging and streaming service.
Integration	Agent-based integration with multiple protocols such as ArtNET/DMX, HTTP, KNX, MQTT, SNMP, WS.
Security	Multi-tenant solution. Multiple users and roles. Access rights: public, private or restricted. Security OAuth. Web component for Identity service.
Asset management	Configuration, Location tracking, status manager.
Infrastructure & Deployment	Cloud and on-premises. Docker images. Hosting as a Service. Edge Gateways on ARM.

Table 5: Mainflux Specifications

Feature	Detail
General architecture	Microservices with high-performance, scalability and fault-tolerance. Domain model: Users, Things and Channels. Scalable NATS broker for internal message exchange.
Data communication and connectivity	Pub/sub multiprotocol messaging bridge, MQTT, HTTP, WebSocket, CoAP, NATS.
Rules and business processing	Does not explicitly feature a rule engine, could be integrated as a custom plugin.
Integration	Third-party integrations with enterprise systems (e.g. ERPs, BI, CRM, SQL/noSQL databases, and other cloud services such as analytics).
Security	Authentication and authorization with API keys and scoped JWT. OpenID Connect with OAuth 2.0. Mutual TLS Authentication (mTLS) with X.509. Nginx HTTPS reverse proxy. TLS and DTLS load-balancing / termination. Vault data encryption.

Edge capabilities	Optimised hardware, Solid Run HummingBoard CBi - Edge1 IoT Edge Gateway. Low memory footprint, low latency, high performance, tolerance for temperature range (-40° to 85° C).
Development & Deployment	Mainflux Security server written in Golang. SDKs and client libraries in C/C++, JavaScript, Python. Microservices containerised by Docker and orchestrated with Kubernetes.

Table 6: Thingsboard Specifications

Feature	Detail
General architecture	Microservice-based instances for ThingsBoard clusters. Gateway for legacy connectivity.
Data communication and connectivity	HTTP, TCP, MQTT. REST API calls, WebSocket subscriptions. Monitor device connectivity states. gRPC potentially shifting to Kafka.
Rules and business processing	Rule engine with event-based workflows. Main components Messages, Rule Nodes, Rule Chains.
Integration	LORIoT, AWS IoT, IBM Watson, Azure IoT, SigFox, and custom.
Security	OAuth2 support, MQTT over SSL. Device authentication: access tokens, MQTT credentials X.509 certifications.
Deployment	On-premises and cloud.

Table 7: Kaa Specifications

Feature	Detail
General architecture	Flexible microservices architecture. Any component is replaceable or customizable.
Data communication and connectivity	MQTT, Sigfox, LoRa, NB-IoT, WiFi, BLE, Z-Wave, 2G/3G/4G/(5G), ethernet, gateways, custom.
Rules and business processing	Structured/unstructured data. Data processing pipelines. Integrated with databases such as Cassandra, MongoDB, InfluxDB, and more
Integration	REST APIs and NATS. Support for business tools like SAP, Salesforce and more. Hardware: sensors, gateways, industrial PLC, wearables.

Security	Device communications are secured with TLS or DTLS. Flexible credentials lifecycle management.
----------	--

Table 8: DeviceHive Specifications

Feature	Detail
General architecture	Microservices.
Data communication and connectivity	REST API, WebSockets or MQTT protocol.
Rules and business processing	Does not explicitly feature a rule engine, could be integrated as a custom plugin.
Integration	REST APIs and NATS. Support for business tools like SAP, Salesforce and more. Hardware: sensors, gateways, industrial PLC, wearables.
Security	HTTPS, WebSockets. TLS for devices and apps. Separate Auth service. Authentication is secured by JSON Web Tokens (JWT).
Deployment	Public, private or hybrid cloud resources. Docker and Kubernetes deployment support. Kafka service communication and load-balancing.

2.2 Commercial perspective

The major commercial IoT platforms included in review are presented in Table 9. These platforms are provided by the biggest digital companies in the world and their feature set evolves rapidly, so the description of the characteristics and features are based on the information available at the time of review.

Table 9: Commercial IoT Platforms.

Name	Description
Azure IoT	Azure IoT is a large and complex end-to-end IoT portfolio. The three most relevant platforms regarding the analysis are Azure IoT Hub, Azure IoT Edge and Azure Digital Twins. IoT Hub is a PaaS message hub for bi-directional communications between an IoT application and related devices. Microsoft also offers Azure IoT Central, which is a fully managed Software-as-a-Service (SaaS) for easy design but less customization. Azure Digital Twins is a platform capable of building digital representation of real things, locations, business workflows, and people processes. The managed services offer a variety of building blocks for building customised solutions in different scenarios [11].

AWS IoT	AWS has broad and deep IoT services, from the edge to the cloud. AWS IoT offers AI integration, meaning efficient model creation and deployment. AWS IoT is built on a secure and proven cloud infrastructure and scales to billions of devices and trillions of messages. AWS IoT integrates with other AWS services and solutions. Based on AWS documentation, its offering is divided in three main categories Device Software, Control Services and Analytics Services [12].
Google Cloud IoT Core	The key Google solution for IoT is Google Cloud IoT Core. Like Azure IoT and AWS IoT, Google Cloud IoT is a managed service that enables secure connection, management and data operations in distributed device configurations. It is easily combined with the functionality of Google Cloud Platform (GCP). Google Cloud IoT Core and GCP offer together world-class ML and big data analytics features for IoT applications [13].
PTC Thingworx	ThingWorx Industrial IoT Solutions Platform promotes a diverse set of capabilities that enable powerful solutions for design, manufacturing, service, and industrial operations. Thingworx has a special focus on Industrial IoT (IIoT), and it provides pre-defined building elements for the IIoT use cases. The pre-configured solutions cut time-to-market, provide flexibility and deliver value rapidly for enterprise customers of all size. ThingWorx offers capabilities that can be divided in five focus areas: connectivity, deployment, analytics, device management and visualization. Their approach is slightly distinct from other service providers, since the product offering is divided. For example, other commercial products often join the connectivity and device management in same elements [14].
IBM IoT portfolio	IBM have three technologies of interest for the IoT domain, IBM Edge Application Manager, IBM Maximo Application Suite and IBM Watson IoT Platform. IBM Edge Application Manager presents an autonomous management solution for edge applications that require qualities such as scalability, variability and evolution. IBM Maximo is platform for asset management, monitoring, predictive maintenance, and computer vision. IBM Watson IoT Platform utilises the AI and cognitive abilities of Watson AI. It included capabilities for real-time data analysis, insight production through AI and ML. Watson IoT can govern applications and device statuses: usage and performance, anomaly detection, and data validation [15].
Cumulocity IoT	Cumulocity IoT from Software AG is built to enable open, rapid deployment and distributed processing. Cumulocity tackles architectural complexity by promoting a singular architecture that covers the infrastructure from edge of the cloud to the on-premises assets. Cumulocity IoT differentiates itself with a special focus on industrial equipment. The focus is highlighted by the support for field-bus protocols such as Modbus, Can bus and OPC-UA. Additional

Particle	<p>distinction is that Cumulocity IoT promotes no-code and self-service approaches [16].</p> <p>Particle offering is a combination of IoT platform and hardware. The product ecosystem of particle includes asset tracking, hardware for prototyping and mass production, development tools and various software APIs and libraries. The functional focus of IoT platform includes IoT devices management, data pipelines, and tracking systems. A special product is the EtherSIM, which is a global, scalable and self-learning cellular connectivity. In the analysis of Task 1.2, the most relevant products are Particle Device OS and Device Cloud. Particle frequently is more data-efficient than its competitors. With the CoAP over DTLS approach, they can hold up connection in worse networks. For example, a MQTT connection over TLS/SSL requires full 5K TLS handshake for a device re-connection [17].</p>
----------	---

Table 10 to Table 16 below discuss in more detail, the characteristics and common features of the platforms presented in Table 9.

Table 10: Azure IoT Hub Specifications

Feature	Description
General architecture	<p>Message broker and micro-service architecture. Device-to-cloud telemetry, Per-device identity, IoT devices and cloud gateways. Lambda architecture (warm and cold paths) for data processing.</p>
Data communication and connectivity	HTTPS, AMQP, MQTT, WebSockets.
Rules and business processing	Features a fully-fledged rule engine allowing for rule-based event discovery and actuation.
Integration	<p>Azure Event Hubs, Azure IoT Edge,</p> <p>Custom third-parties connectors such as Thingsboard and ThingWorx.</p>
Security	<p>Device Provisioning Service (DPS) and authorization. X.509/TLS-based Handshake and encryption. Azure Active Directory for IAM. Key Vault. Policy-based access control.</p>
Infrastructure and deployment	Public, private or hybrid cloud resources. Docker and Kubernetes deployment support. Kafka service communication and load-balancing.

Table 11: AWS IoT Core Specifications

Feature	Description
General architecture	Publish/subscribe message broker
Data communication and connectivity	MQTT, MQTT over WSS, and HTTP, LoRaWAN
Rules and business processing	MQTT topic stream for rule analysis and execution. Passed through the pub/sub broker with device communication protocols or Ingest feature to optimise data streams by deleting the pub/sub message from the ingestion path).
Integration	Amazon Sidewalk, Alexa Voice Service, other Amazon services such as AWS Lambda and S3.
Security	IAM roles and policies. Amazon Cognito Identity. AWS security credentials. Authentication at TLS layer with X.509 certificate chain. Support for custom authentication. Data protection with multi-factor authentication, SSL/TLS for communication with AWS resources, logging with AWS CloudTrail.

Table 12: Google Cloud IoT Specifications

Feature	Description
General architecture	Publish/subscribe architecture, complemented by GCP pub/sub features. Main components are device manager and protocol bridges for MQTT and HTTP
Data communication and connectivity	HTTPS, MQTT. Gateway (over the HTTP and MQTT) support for example ZigBee and Bluetooth.
Rules and business processing	Does not feature and explicit rule engine.
Integration/interfaces	Google Big data analytics and ML services, BigQuery, Bigtable, and thirdparty services.
Security	JWTs for short-span authentication between devices. Public/private key authentication for verifying device credentials over TLS 1.2. IAM with Cloud Identity, Google Cloud's built-in managed identity. Two-factor authentication.
Infrastructure and deployment	AWS cloud environment. AWS IoT Device Client, a device-side reference implementation. AWS SDK and AWS IoT.

	Device SDK provide APIs and tools. Google Cloud. REST APIs for device registration, deployment, and operations.
--	---

Table 13: PTC ThingWorx Specifications

Feature	Description
General architecture	ThingWorx Connection Server, a monolithic server application for device connections and message routing.
Data communication and connectivity	HTTP, SOAP, SQL, OData, Swagger, RAML, and OSLC.
Rules and business processing	Offers the tools to create and customize instances of "ThingWorx Business Rules Engines".
Integration	ThingWorx Flow: Microsoft Azure, SAP, Salesforce, Windchill, Box. Separate Azure IoT Hub Connector for Azure IoT Edge.
Security	Encrypted communications with SSL and TLS. Authorization with agent-based communications. Support for plug-in authentication, such as LDAP, active directories, and custom solutions.
Infrastructure and deployment	On-premise servers, cloud resources, and hybrid environments. ThingWorx offers role based and user-friendly IoT web applications. Users can view, analyse, and react to IoT data in real-time.

Table 14: Particle Specifications

Feature	Description
General architecture	Pub/sub architecture. Webhooks.
Data communication and connectivity	Cellular 2/3/4G, Wi-Fi REST APIs, CoAP. Server-Sent-Events: Heroku, Amazon EC2, Google AppEngine etc.
Rules and business processing	Features a Node-RED-like interface for creating and managing rules.
Integration/interfaces	GCP, Azure IoT Hub, InfluxData, QuestDB.
Security	RSA public-key pairs. DTLS over UDP, or AES over TCP.
Infrastructure and deployment	Particle's hardware, Device OS and Device OS OTA updates.

Table 15: IBM IoT Specifications

Feature	Description
General architecture	Publish/subscribe-like Broker.
Integration/interfaces	IBM Cloud services such as Db2 Warehouse, Cloudant NoSQL DB, Object Storage, Secure Gateway, or non-IBM services with custom interfaces.
Data communication and connectivity	HTTPS, MQTT, custom protocols.
Security	Platform Service manage device connections and security. OAuth server. Data security and integrity are embedded in the IBM Cloud storage solutions.
Infrastructure and deployment	IBM Cloud. APIs to manage device registration, deployment, and operations. Distinct Analytics and Blockchain services.

Table 16: Cumulocity IoT Specifications

Feature	Description
General architecture	Agent-based model Applications are either web-based UIs or server-side microservices.
Integration/interfaces	webMethods.io Integration, cloud integration solution from Software AG. Through webMethods.io possible integration examples are Marketo, Salesforce, Gmail and more.
Data communication and connectivity	MQTT (over WebSockets support), HTTP, REST, Modbus, Can bus and OPC-UA.
Security	Encrypted communications. HTTPS connections from devices to applications.
Infrastructure and deployment	Standard tenants hosted at AWS. Custom hosting depending on the subscription. Microservices deployed as Docker images.

2.3 EU research perspective

IoT-NGIN draws information from the EU research projects and applies it in various formats. The research projects develop state-of-the-art technology that is not feasible for smaller open-source projects or is not yet commercially exploitable. The researched technologies might still have substantial effect on the future development of IoT. The IoT European Large-

Scale Pilots Programme (LSP) consist of several innovation consortia that conduct research and development in the IoT domain.

The projects work in IoT integration across value chains, enhancing scalability and focusing on use-case contexts. Demonstrating in operational environments is also a crucial part of the Programme and significant part of the projects listed in Table 17.

Table 17: European Research Projects relevant to IoT-NGIN.

Project	Description
CREATE-IoT [19]	Cross fertilisation through AlignmentT. CREATE-IoT focuses on the synchronisation of projects and technologies, with an additional focus on IoT Exchanges. CREATE-IoT aligns the innovation ventures with, for example, the Alliance for Internet of Things Innovation (AIOTI). In general, the objective is to advance collaboration between IoT initiatives and promote IoT ecosystems based on open innovation.
SynchroniCity [20]	Delivering an IoT-enabled Digital Single Market for Europe and Beyond. SynchroniCity research on reference architectures and a framework regarding “minimal interoperability mechanisms” or MIMs. The MIMS take a neutral stance on service providers and technologies. The project currently revolves around three MIMS: context information management (CIM), common data models, and ecosystem marketplaces. Future development includes concepts such as personal data management and fair artificial intelligence.
MONICA [21]	Management Of Networked IoT Wearables – Very Large Scale Demonstration of Cultural Societal. MONICA demonstrates large scale IoT in the domain of smart living. The technologies are using wearable and interoperable sensors. The things are embedded into cloud platforms that provide a variety of functionalities to build applications.
AUTOPILOT [22]	AUTOMated driving Progressed by Internet Of Things. AUTOPILOT develops new services and models for the mobility industry. The new applications use autonomy as the base principle and build on top of autonomous vehicles. Other autonomous driving applications such as automated parking and dynamic digital maps.
ACTIVAGE [23]	ACTivating InnoVative IoT smart living environments for AGEing well. Effective use of existing technology is extremely important for large-scale applications. Similar to IoT-NGIN, ACTIVAGE reuses and scales open-source and proprietary IoT technologies. In addition, it develops new interfaces to integrate the existing solutions and provide interoperability between systems in smart living, especially in health technology related to activity and ageing.
IoF2020 [24]	Internet of Food and Farm 2020. IoF2020 focuses to enlarge IoT adoption across the farming and food value chains. Sufficient, efficient, and safe delivery of healthy food can be secured with the newest technologies, even in harsh conditions. The project exploits a symbiotic ecosystem of farmers, food producers, technology vendors,

	and academia. IoF2020 is based on open architecture and infrastructure bases on reusable elements, following the latest standardization and security concerns.
GAIA-X [25]	GAIA-X is set to be an Infrastructure and Data Ecosystem according to European values and standards. The core concepts of GAIA-X are Digital and Data Sovereignty. In the GAIA-X project, the concept of digital sovereignty is defined as "the power to make decisions about how digital processes, infrastructures and the movement of data are structured, built and managed". GAIA-X has three main technical approaches: (1) Federation, (2) Self-Description and Policies, and (3) Identity and Trust. IoT-NGIN closely aligns with the concepts and aims to implement similar high-level objectives, all according to the European Strategy for Data [26].
AGILE [27]	AGILE (Adaptive Gateways for diverse multiple Environments) is a consortium that builds an extensible and versatile IoT gateway. The gateway supports interoperability, IoT device and data management, diverse IoT applications, and interfaces for third-party cloud services. The most important design principle is modularity. AGILE is open-source, and it will be embedded within an IoT project of the Eclipse Foundation [28]. AGILE has an emphasis on new developer communities. The objective is that an intense collaboration leads to agile prototyping opportunities and ultimately increases the adoption rate of the project.
U4IoT [29]	User Engagement for Large Scale Pilots in the Internet of Things. The objective of U4IoT is to build tools for the LSP projects for user adoption and engagement. The tools include online resources, crowdsourcing, various best practices and a privacy themed game for learning. In addition, U4IoT examines societal, ethical and ecological perspectives and makes recommendations on how to tackle barriers that hinder IoT adoption. Education and sustainability requirements are along with the key obstacles.

In addition to IoT projects listed above, some other EU research project might not be directly related to IoT but there is a significant and interesting relationship between them and IoT-NGIN. Those projects and their relation to the IoT-NGIN have been further analysed in Table 18 below.

Table 18: IoT-NGIN relevant research projects and their relationship with IoT-NGIN.

Project	Relation to IoT-NGIN
SOFIE [30] SOFIE (Secure Open Federation for Internet Everywhere) develops a federated blockchain platform. The platform offers integrity, confidentiality and auditability of	IoT-NGIN can adapt data privacy and data sovereignty through the federated blockchains and ledger-independent transactions of SOFIE. IoT-NGIN can then share its concepts and tools, digital services data in a secure and open way.

<p>data and actions through ledger-independent transactions.</p> <p>NRG-5 [31]</p> <p>NRG-5 is a leading project on the energy vertical of 5G PPP/5G initiative. The project advances the path of energy producers and service providers into decentralised energy systems and renewable energy, paired with preventive maintenance.</p> <p>OPEN DEI [32]</p> <p>OPEN DEI project contributes to the EU Digital Transformation strategy by providing synergies, best practices, and increased collaboration of regional and national actors. OPEN DEI takes part in creating common data platforms through unified architecture design and applying the latest industry standards.</p> <p>SOGNO [33]</p> <p>SOGNO is developing new cloud-native technology for data-intense energy and control systems. It exploits concepts such as low-cost hardware, ML and 5G. In parallel, RESERVE develops frequency and voltage control for renewable energy.</p> <p>INTER-IoT [34]</p> <p>INTER-IoT focuses on reference meta-architecture and reference meta-data model to form guidelines and base for interoperability system across IoT use cases. The designed reference model and architecture are both based on the Lighthouse project IoT-A. INTER-IoT introduces Functional Groups and Functional Components as means to define relevant structure and details. Examples of the Functional Groups are Security, Communication and IoT Service. Each Functional Group consist of more detailed Functional Components [35].</p> <p>AI4EU [36]</p> <p>AI4EU promotes a sharing platform for various AI assets, such as high-level services,</p>	<p>IoT-NGIN benefits from NRG-5 expertise in 5G and adapts the learnings in testing setups and new services.</p> <p>IoT-NGIN harnesses the experiences of Open DEI work on digitalisation of the energy vertical. Also, OPEN DEI is a project partner on data sovereignty with International Data Spaces Association (IDSA) which benefits IoT-NGIN.</p> <p>Service implementations in IoT-NGIN based on the SOGNO approach of low- cost hardware, ML and 5G. The insights of RESERVE project lay out the basis for electric vehicle charging optimisation and smart grid use case requirements.</p> <p>INTER-IoT provides a reference model, meta data model and reference architecture that align with the research interests of IoT-NGIN. The structure of INTER-IoT inspires meta-architecture development.</p> <p>IoT-NGIN applies the existing knowledge in federated ML and data privacy. In addition,</p>
---	---

components, learning and testing datasets and high-speed computing resources. The platform is based on the Acumos AI platform developed by the Linux Foundation.	partners involved in both projects can bring insight to running use case pilots.
--	--

3 IoT architectural patterns overview and analysis

This section provides a list of identified IoT architectural patterns, combining the ones identified in the review phase described in Chapter 3 and the new IoT-NGIN specific ones. IoT-NGIN provides unique addition of patterns such as Distributed Ledger Technologies (DLTs) and 5G capabilities in IoT communications. These patterns are not meant to be exclusive or exhaustive. Different patterns can be combined in actual implementations to achieve a solution for a product or service in questions. Also, as technologies and solution architectures evolve, the meta-architecture will accommodate these changes in future versions.

3.1 Layered architecture

Layers of isolation are an abstract architectural concept similar to separation of concerns in software engineering. The core idea is that changes in one layer do not affect other layers. Even in the worst case, the changes should be isolated to associated layers. As an abstract concept, layer is a selected analysis viewpoints and different reference architectures consists of different layers. An example of layering is presented in Figure 2.

Each layer should be independent and handle other layers as "black-boxes", only having the information about the required interfaces and data formats. For example, the isolation concept allows to switch between different protocols for increased performance in network layer, but the change does not affect higher level services or the user interfaces.

Overall, the layered architecture pattern is a valid general-purpose solution and serves as an excellent starting point for many applications. One of its benefits is that other more specific patterns can be attached into the layered model, making it easier to derive detailed conclusions and actions. The downside of the pattern is that it tends to point towards monolithic applications [37].

3.2 Microservices-oriented architecture

Microservices architecture is a common software architectural pattern, often interpreted as part of service-oriented architecture (SOA). Today however, microservices are such a dominant pattern in software and system development that it is interpreted as distinct from SOA.

Microservices are independent services that can operate on various system levels. Each of these has its specific purpose or task and microservices can be seen as single self-contained deployable entities. Complex systems can thus be built by composing a multitude of such services (Figure 3). The service should then only serve a single functionality with a defined interface. This has the practical impact, that a small development team can create and maintain such a service with only minimal dependencies to other business units. This also alleviates the replacement of such services with a better-suited implementation without affecting the whole system.

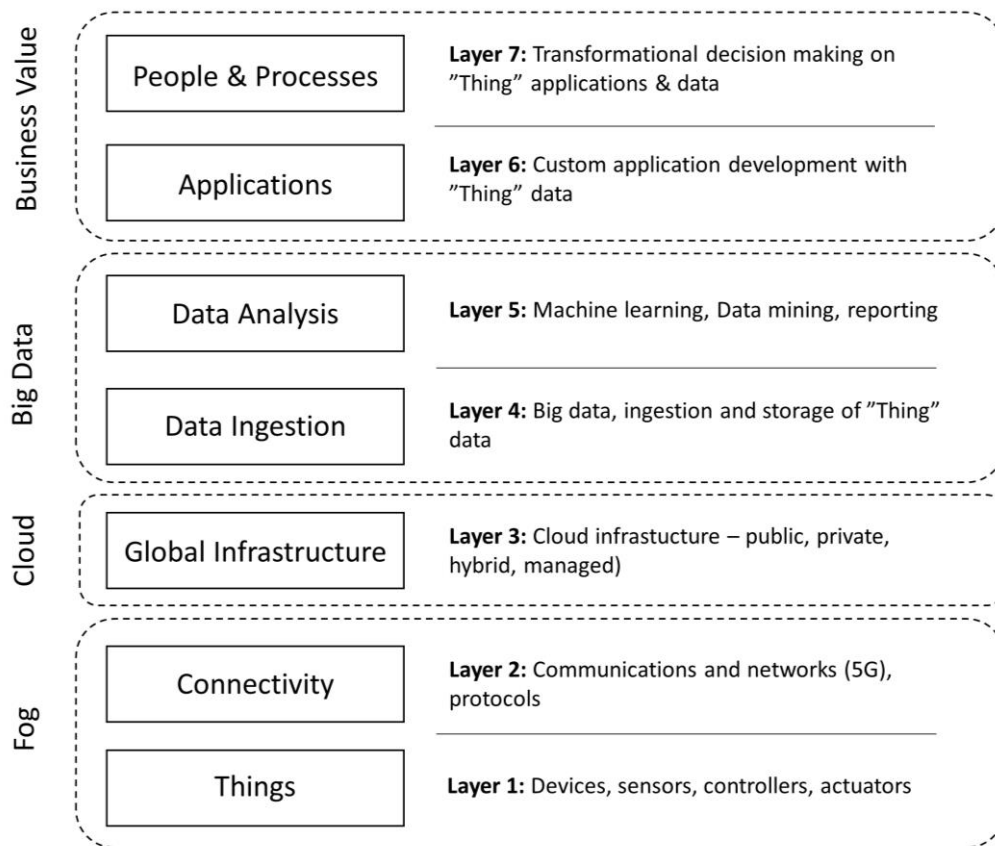


Figure 2: IoT Layers [38].

The requirement for self-containment usually requires the service to be stateless as well. Shared state is achieved via databases, which are connected to the services. Communication with microservices often happens through APIs or message queuing based on common protocols such as http or mqtt.

This architecture pattern can enable massive horizontal scaling, as the services can run on (even locally) independent hardware. Load balancers and content delivery networks can effectively distribute the load onto the services, which can also be dispatched quickly to react to variances in demand.

On the other hand, the encapsulation increases the complexity, which this makes debugging, and system overview harder. Managing a scalable deployment of a system build from a multitude of independent services is therefore not an easy task. As the services communicate via network protocols, a reduced performance and increased latency compared to monolithic software has to be considered.

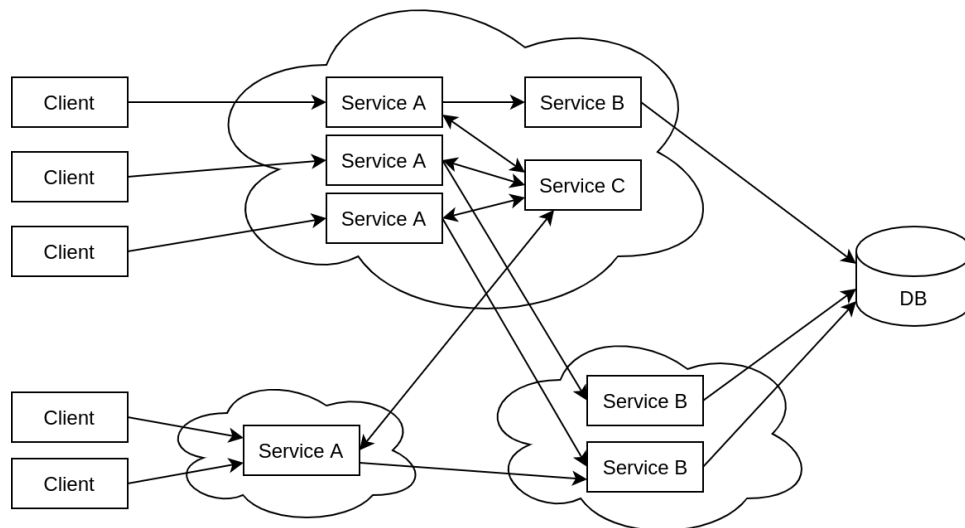


Figure 3: Microservices Architecture.

Comparing to the layered IoT architectures, microservices can be discovered from almost every layer, but are often part of the core application infrastructure and deployed in cloud environments. For example, AWS facilitates architecture migration to cloud-based microservice infrastructure. Concurrently, the wide adaption of Linux containers and virtual machines enable to maintain customised execution environments. The microservice development has also paved the way for different software deployment services, such as Docker and more high-level container management systems such as Kubernetes.

3.3 Event-driven architecture

The entire IoT industry is built around different events, action triggers and asynchronous communication. Event-driven architecture embeds the characteristics into an architectural pattern that offers scalability and flexibility. The building blocks of event-driven architecture are distributed, single-purpose components that produce, process and receive various types of events.

The pattern frequently builds on top of two core elements, mediator and the broker. Mediator concept is used in applications that require complex process steps or management that benefits from central event processing. It can implement supporting components such as event ques or event processing pipelines.

Yet, the communication between the services and applications is more often performed by a broker (or a message broker). Often the message broker implements a pub/sub architecture, which is seen across the open-source and commercial solution. The broker handles event-based requests from connected sensors, devices and other sources and vice versa, enabling transmitting actions and messages back to the event sources. Event chaining in the message broker then orchestrates the message to the right microservice, function or interface. Feasible implementation models include centralised and federated designs. The broker also contains required event channels that handle the data transfer and information flow. The event channel can consist of message queues, message topics, or both. MQTT has

become almost the de facto standard in message brokering and it is implemented in various services [37].

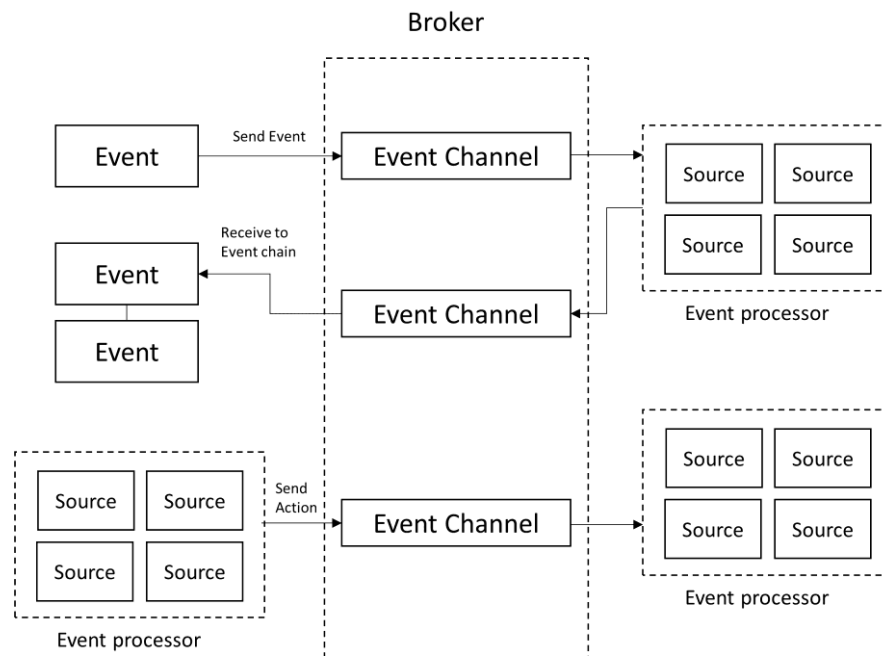


Figure 4: Event Broker components.

3.4 Lambda architecture

Many of the IoT-related use cases share characteristics that drive the need for new data processing capabilities and components. Lambda architecture is new architecture that consists of real-time data processing and batch data processing. The capabilities of Lambda architecture include:

- Process streaming data generated by sensors and edge devices.
- Scalability and elasticity: handle a growing number of edge deployment reliably, in real-time or in frequent batches.
- Enable storage and management for big data, enable pattern analysis and selection of optimal machine learning models.
- These requirements differ from the requirements that drove the development of traditional data warehouses based on relational database technologies.

The division into speed and batch layers and focus on the data processing is distinctive for this pattern. The concrete lambda architecture consists of various typical IoT components, such as devices, gateways and event hubs. The pattern is subject to different variants and selecting the specific elements depends on the domain and environment. Existence of legacy components and limited network coverage. It is important to note that often the elements are architectural or design patterns themselves [39]. Figure 5 illustrates the elements of a Lambda architecture.

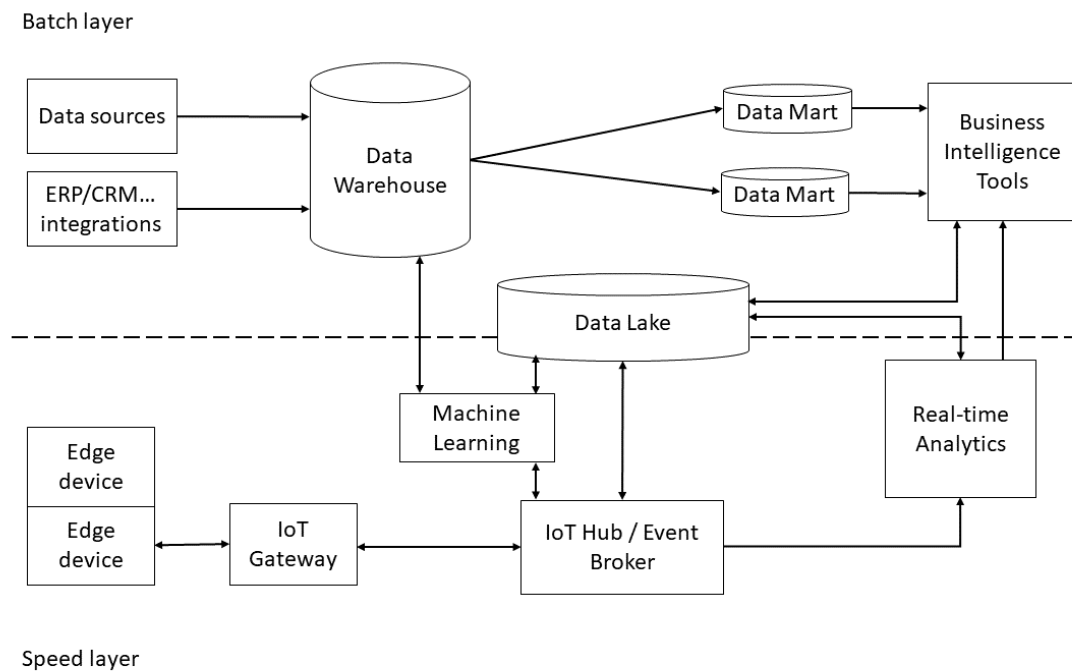


Figure 5: Lambda architecture elements

In the context of IoT-NGIN, edge devices are examined as the key component in architecture pattern. Gateways and Hubs (event-driven brokers) are also stand-alone patterns. In addition, Machine Learning (ML) in IoT is often analysed either as a separate domain itself, or as a specific design pattern of architectural patterns.

3.5 Software-defined networking (SDN)

SDN is a solution for effective network management and deployment that is based on a centralised network perspective. The number of stakeholders and users in the core network infrastructure is increasing alongside the growing number of heterogeneous devices in the network. SDN provides network operators and users with new resource control methods that help to develop use case specific solutions.

SDN focuses on separating the responsibility of the control plane from the data plane, enabling custom and real-time configuration of network control at the device level. A general view to SDN includes infrastructure, control and application layers. The layered architecture is complemented with API definitions of northbound and southbound APIs. Control and application layer are interfaces with the northbound API, and the southbound API connects the infrastructure and control layers. Eastbound and westbound APIs do exist for interfacing between network controllers, but these horizontal APIs are rarely used in architecture design.

SDN embeds well to the general layered-architecture pattern, and it includes monolithic tendencies since its object includes enforcing centralised control. However, in the network infrastructure context SDN is already being applied in the ultra-distributed edge cloud

environments because it provides excellent flexibility and manageability. SDNs can also be leveraged at the aggregation of unstructured data, global network monitoring, optimizing the efficiency and management [40].

3.6 Edge computing

Edge computing architecture consists of networking hardware, edge devices, and client software that power computing and processing at the edge of a cloud. The edge resides in between centralised servers and end-user clients. The main advantage of edge computing is that data is processed at the network edge, and more centralised cloud server infrastructure, e.g. data center only receive the processed data. Transferring data back and forth causes a heavy load to the network and introduces security and privacy concerns. Thus, the approach will save network bandwidth and reduce the amount of energy lost in the transmission.

The distributed nature of the edge brings it close to the end-user and can provide lower latency because the time to access data storage and perform data transfer is shorter. Edge computing can also ensure an increased level of end-to-end security since it has more control over the device connectivity, which is often vulnerable to several types of attacks. In general, edge computing is a more potent option for IoT applications than traditional centralised cloud computing services. The largest cloud vendors have identified the current development and most of them offer mature edge computing capabilities. Valid examples include AWS IoT Greengrass, IBM Edge Application Manager or Cumulocity IoT.

There are other perks such as location awareness that can be built into the edge application. Since the devices are located on the network edge, the applications are already aware of the location information and context, which makes development easier. There is also a rapid need for extensive, high-performance and application specific IoT infrastructure. Network infrastructure and edge computing paradigms are at the center of solving the challenge. In parallel with 5G development, network management needs to be further developed. In this context, software-defined networking (SDN) and network function virtualisation (NFV) are potential concepts. To summarise NFV, it can offer tools for edge implementation in distributed or constrained conditions. Through resource virtualisation, NFV provides mechanisms for mobility management, device authentication, fault-tolerance, and management of data and mobility [41]. NFV is linked closely to IoT-NGIN, and it is presented as a key pattern in the meta-architecture of the project.

3.7 Wireless sensor networks architectures

Wireless sensor networks (WSNs) are built out of compact, portable and power-restricted sensor nodes. In the IoT domain, the sensor nodes are the connected “things”. The nodes include information that can be transmitted across radio links and gateways into various locations, Internet servers, or client devices (Figure 6). The sensor networks are the base for many applications that include interaction between an environment and user through any kind of sensor or actuator. WSN development includes work in sensor, microcontroller and transceiver technologies. It inspires a set of real-time application scenarios such as environmental control, military surveillance or healthcare systems.

The sensor nodes can monitor the collected data that is then transmitted forward with actions called hopping. The prevalent communication mechanisms include single-hopping (or direct hopping) and multi-hopping. In multi-hopping, sensor nodes are not required to establish direct communication with base stations. Rather, the data is transmitted in several steps, reducing the responsibility of a single node which in turn results in a more stable network connection. Multi-hopping is more power-efficient than single-hopping and offers more potential for IoT applications [42].

Design alternatives for WSNs are proactive or reactive communication. For a proactive approach, sensor nodes are continuously listening and periodically sending signals, regardless of the actual need. The proactive approach requires more power and is not as efficient as the reactive mode, which only reacts to certain predefined events and signal thresholds. The notion of proactive and reactive WSN design resonates well with the development of microservices. The proactive mode can be linked to more complex applications through analogies of event-driven architecture, message brokers, and NFV.

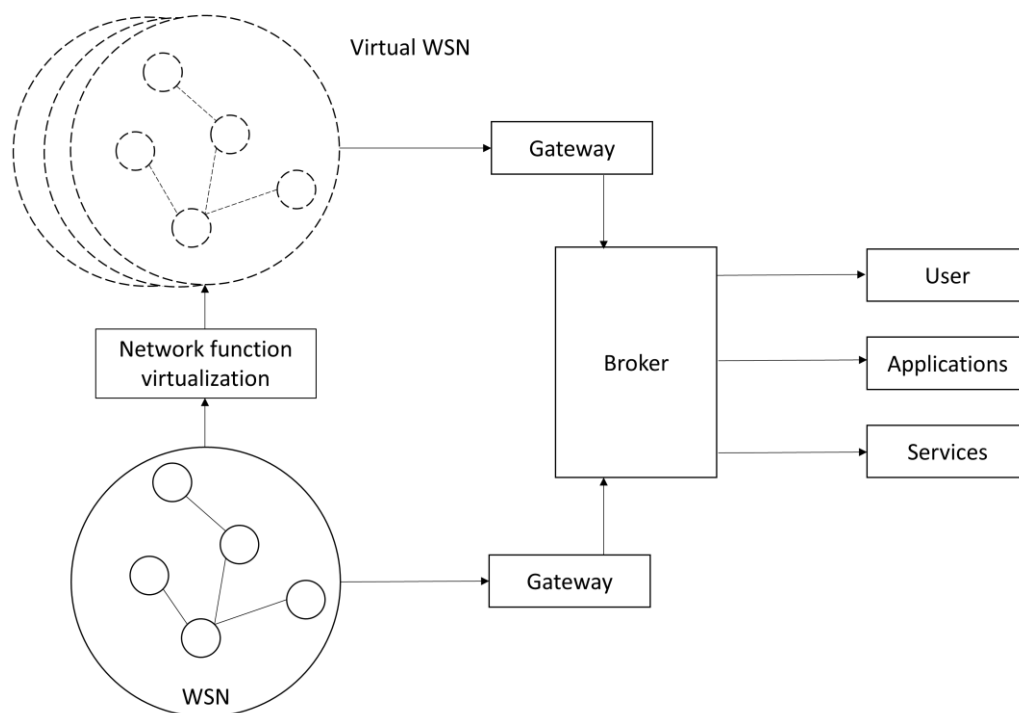


Figure 6: Wireless Sensor Networks overview.

3.8 5G network exposure

Increasing digitalisation is occurring in many vertical sectors. For example, industry manufacturing processes need to be automatised in order to increase productivity and reduce delays/faults. Besides, the consumers require improved user experience for online gaming. These and other use cases can be enhanced by exposing 5G resources.

Exposure means that new applications will benefit from the abstraction of underlying resources. In this way, the complexity of telecom networks can be hidden, and application

developers can focus on the application logic development, which makes it possible to bring applications or new services faster to the market. It is foreseen that edge computing applications will especially benefit from the 5G resource exposure. The following 5G resources can be exposed to non-telco applications and services:

- 5G network exposes connectivity and mobility network capabilities traditionally available to the network operator solely.
- Communication services such as messaging, audio and video calling can support services in different verticals.
- Operational and business support system functions that handle operational and commercial management aspects of exposed network services and capabilities can be exposed to applications.
- Telecom networks provide runtime execution environments that may host virtual network functions (VNF) and non-telco applications. Accordingly, applications may require different edge computing platform elements such as cloud computational, networking and storage resources.

3.9 5G network slicing

5G network slicing is a critical IoT-capability that enables optimisation of IoT and 5G networks. SDN and NFV are important concepts that support scalable and flexible network slicing.

The Network Slice Management is done through an Application Function, which in 3GPP specifications is called Network Slice Management Service (NSMS) or Network Slice Subnet Management Service (NSSMS) that support different use cases defined in TS 28.531. A slice subnet is considered different segment of the end-to-end system e.g. Radio-access network (RAN) subnet, Transport subnet (TS), Core subnet (CN). NSMS provides an interface to create, activate, terminate slices in the network and perform feasibility checks before creating new slices. The NSMS will interact with different modules in order to create, delete and modify the network slices. Each subnet module (RAN, TS, CN) will utilise different technologies such as CN might utilise network orchestrator based on OpenStack and Kubernetes to create new 5GC instances.

The NSMS will allow the network operator to create slices for isolating IoT-NGIN traffic and assign different resources. 3GPP has defined in TS 29.641 the data structure to define the features supported by the slice. Moreover, the QoS and other parameters that will require resource allocation will be defined when creating the slice as part of the slice data structure. Thus, the slice information is maintained by the data structure that includes nested structures described in TS 28.541 that includes nested and inherited data structures as shown in Figure 7 [43].

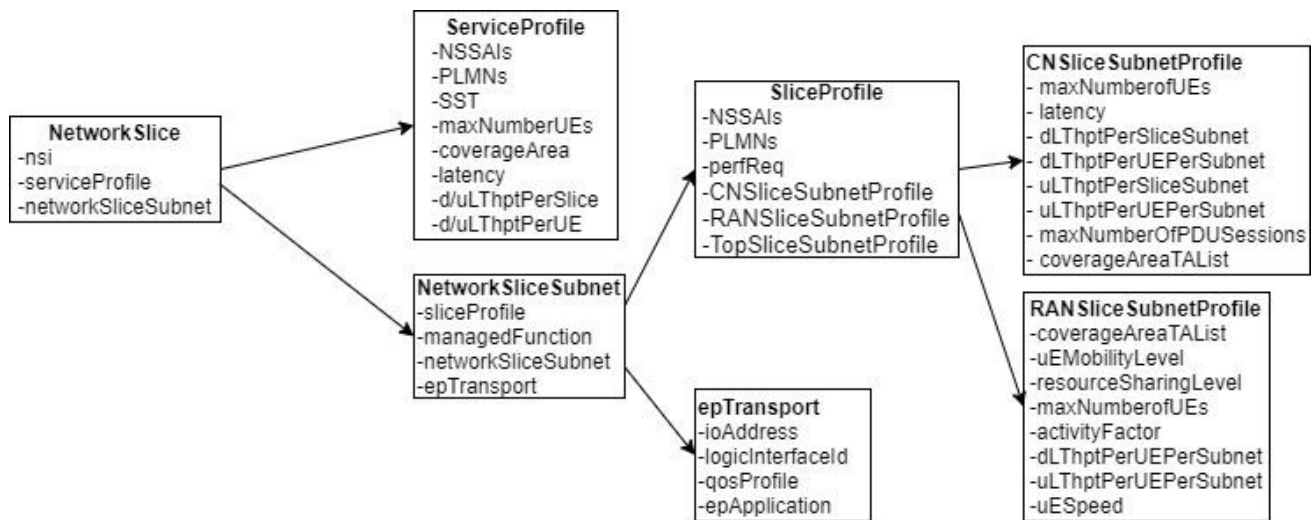


Figure 7: 5G network slicing structure.

3.10 Gateways

IoT gateways are closely linked to edge and cloud computing and enable communication between sensors, controllers, devices and cloud. Edge devices can be labelled as field gateways. The field gateways are specialised devices or general-purpose server hardware. Responsibilities of field gateways include device control and management of data processing and M2M communication. In general, gateways act as a building block and other devices can be attached to it to build larger systems. A field gateway is often located in a fixed environment and close to sensors and actuators. There is a distinction between traffic routers and field gateways since the gateway actively manages information processes in the network [44].

Field gateways can connect to external surfaces, often through cloud gateways. Cloud gateways orchestrate the two-way communication from devices and edge to the wider network infrastructure. The network space can be private or public, depending on the application. The cloud gateway can be a distinct physical device, or a virtual version embedded in the cloud infrastructure. In this context, the cloud is perceived as the distributed computing, data storage, and network infrastructure that is not in the immediate reach of the operating environment and independent of the connected devices. NFV can be applied for cloud gateway to isolate the cloud gateway and connected entities from other network traffic, thus increasing security and privacy.

Gateways enable connectivity through standard and custom protocols, including MQTT and HTTP, Zigbee or Bluetooth. Many platforms support design with gateways, for example, OpenRemote and Azure IoT. Mainflux offers its own MFX-1 IoT Edge Gateway. Gateway technology is also an ongoing research interest to many projects, such as AGILE.

3.11 Distributed ledger technologies (DLT)

Distributed ledger technology (DLT), such as blockchains, is an emerging paradigm that is getting increasing public and research attention. DLTs form distributed peer-to-peer networks in which non-trusted users are able to interact and perform transactions without controlling

intermediaries. A number of blockchains, especially cryptocurrencies and other digital assets are based on public ledgers, but also private ledger services developed by enterprises do exist. DLTs are not yet widely adopted in the IoT industry, but the research of the topic is accelerating as the technologies mature.

Then main features of DLTs include data immutability, distributed consensus mechanisms and data replication. Public DLTs are, in general, secure and offer high availability. It is important to note that DLTs have a lot of variances in terms of achieved functionality. DLTs take differing approaches on the technology and common trade-offs include latency, type of consensus algorithms, security, and performance (transactions per second). An interledger or inter-DLT method can then be helpful in building complex systems as it allows data exchange between DLTs. Multiple (different) DLTs, each with their own strengths, can then be used together in a single system, thus improving the security, performance, and other properties of the system. Hash-locks and time-locks are among the cryptographic mechanisms that can be used together with the interledger approach to ensure that operations successfully complete in all the linked DLTs. As effective M2M communication is becoming more common along 5G and autonomous devices, DLTs offer potential in securing and federating transactions performed by heterogeneous smart devices [45].

3.12 Serverless computing

Serverless is a computing and architecture model where compact pieces of code are executed in a cloud environment without the need to control the resources on which the code runs (Figure 8). The executed application logic can include functions such as are data ingestion, queries, device updates, or message alerts. The computing runtimes are often referred to as Function as a service (FaaS). Despite the name, serverless does not indicate an absence of servers. It focuses on the fact that operational concerns such as resource provisioning, monitoring, maintenance, scaling, and fault-tolerance are sourced to a cloud provider. Serverless architecture is compatible with many other patterns, such as event-driven architecture and lambda pattern features such as stream processing [46].

Serverless computing is not yet widely adopted across the analysis perspectives. For example, successful open-source projects focusing on serverless are infrequent. One explanation is that efficient and scalable serverless computing requires a vast amount of infrastructure resources. The largest cloud providers are competing on the domain with AWS Lambda, Azure Functions, and Google Cloud Functions. It is often more cost-effective to design solutions based on proprietary solutions with the necessary tools. For example, AWS Lambda implements functions that can run special runtime environments in language customised containers. Competing with highly refined computing resources such as AWS Lambda is difficult, but many of the approaches can be adopted in custom solutions.

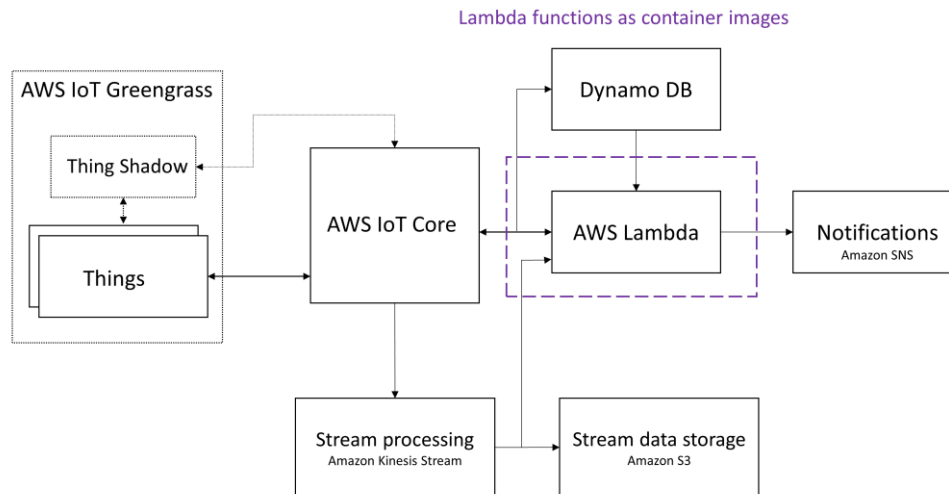


Figure 8: Serverless computing architecture.

3.13 Self-sovereign identities (SSI)

The goal of the self-sovereign identities [47] is to give user control over its identities, allowing identities to be generated without a reliance on the external party. This allows the users to generate and use the identities without any external party monitoring this activity over multiple service, a huge improvement to the privacy of the users. Decentralised Identifiers (DID) are self-sovereign identifiers which are user generated and usually derived from the user's public key. Verifiable Credentials (VCs) are related technology, which allows expression of claims about the subject, in a similar manner as the traditional attribute certificates. In contrast to traditional identifier and certificate solutions, however, DIDs and VCs emphasise self-sovereignty and are machine readable, allowing automation.

3.14 Data federation

IoT-NGIN federation methods enable on-the-fly adaptation and processing of heterogeneous data and control messages. The federated approach for secure cloud framework and on-device ML processes are a core part of the federation activities. In general, only necessary communication data should be transmitted and the original sensor and source data is held at local data storage. Secure, trusted, and open data sharing will be achieved through Inter-DLT technologies. In addition, zero-knowledge techniques for ML models verification without conveying any information apart from the model data ownership.

3.15 Digital twins

The Digital Twin Consortium (DTC) defines a digital twin as: “A digital twin is a virtual representation of real-world entities and processes, synchronised at a specified frequency and fidelity”. There is no exact definition or consensus on what features such a digital twin should entail. The features offered by a specific digital twin implementation is largely use case dependent. The definitions provided by different standardization bodies also reflect this. The Industrial Digital Twin Association defines digital twins as: “A digital twin is a digital representation of a physical or nonphysical asset or process from the real world in the digital world”. In practice this could mean any information including, but not limited to, physics-based models, analytical models, time-series data and historians, transactional data, visual models, and computations [48].

A collection of typical features that make up a digital twin is given in Figure 9. Each of these features may be implemented as its own component i.e., a microservice or a monolithic application. It can be argued that the core problem involving digital twin systems is the question of how to efficiently link these components together to facilitate the creation of a distributed and decentralised system of interconnected digital twins which would support new use cases enabled by advances in 5G, IoT, and ML/AI technologies.

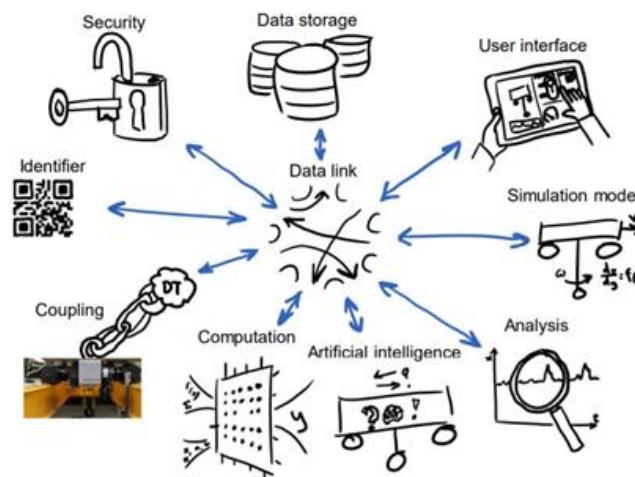


Figure 9: Digital Twins Architecture [49].

IoT-NGIN is taking a meta-level digital twin approach to this problem which can be deployed on top of existing digital twin implementations. The additional meta-level layer abstracts the implementation details, providing a unified interface for higher level operations, for example applications utilizing decentralised security aspects such as DIDs and verifiable credentials. Similar ideas can be found in existing ‘digital twin’ standards such as the Asset Administration Shell (AAS) and the Web of Things (WoT) Thing Description (TD).

3.16 Reference models

Reference models or reference architecture is a high-level architectural abstraction that includes essential building blocks and design rationales. The reference architecture should link functional requirements, quality attributes, and the design activities in system development, standardization, interoperability, and architecture evolution. An example of IoT Reference Architecture is presented in Figure 10.

Meta-architecture is highly related to the concept of reference architecture. The key difference is that as meta-architecture is a second-order meta-framework, it builds out of architectural patterns, instead of design patterns. Including the design patterns in the development process lowers the abstraction level by one. When architecture design shifts focus to the implementation of functionalities, the meta-architecture can transform into a reference model. The meta-architecture includes conceptual analogies from reference architecture design, such as developing different functional groups and sets of software, hardware, and system components [50].

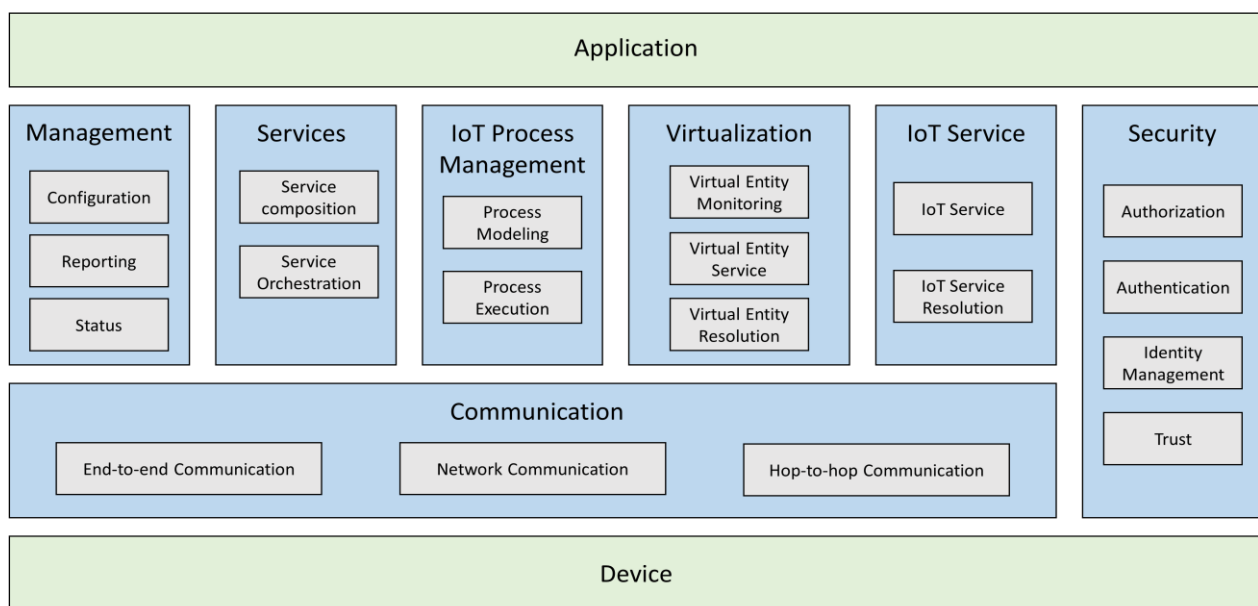


Figure 10: Early model of IoT ARM and its functional view [51].

4 The IoT-NGIN meta-architecture

The aim of meta-architecture work in IoT-NGIN is to provide a research-informed framework for designing and implementing IoT solutions in different usage scenarios. The meta-architecture collects together key quality requirements, architectural patterns and high-level system components. The meta-architecture provides an overall framework for individual system or solution architectures realised in actual implementations.

The IoT-NGIN meta-architecture is based on analysis of current IoT platforms and standards. The analysis of current approaches covered key open-source and commercial IoT platforms and was further extended by EU research perspective covering most relevant EU research projects in recent years. The architectural patterns included into the meta-architecture are based on expert advice and knowledge identified in previous open-source, commercial and EU research projects complemented by information provided by IoT-NGIN project partners.

4.1 Definition

An architecture of a system is the structure set required to proficiently reason about the system. The structures consist of elements, relations, interactions and properties of the system under study. Traditionally, in software engineering architecture refers to software systems and its elements. Architecture possesses a vital role in complex system development and in the system's capability for inter system communication or to change its hierarchy.

The meta-architecture takes a higher-level view and collects together architecturally significant choices, patterns, components, viewpoints and quality attributes that need to be considered when designing and implementing individual systems. The meta-architecture provides a foundation for the architecture strategy and system design. In a sense, a meta-architecture can be considered as a strategic architecture construct that acts as a starting point for architecture design, aiming at reducing the overall complexity of the process, while hinting over good design principles.

The *meta* prefix indicates an architecture of a second order or kind, namely an architecture of architectures. Thus, a meta-architecture is a living collection of architectural patterns, that are paired with expandable application domains and related quality attributes. In other words, a meta-architecture is an orchestration of architecturally significant approaches. Derived concrete technical architecture instances must conform with the principles, constraints and elements of the meta-architecture and do not need to be mapped explicitly.

4.2 Key meta-architecture viewpoints

The four aspects characterizing meta-architecture are scalability, openness, security, and monetization:

- **Scalability:** Standalone IoT-focused 5G optimisation through a secure edge cloud micro-services platform achieves scalability by design. Secure by design federation of communications and data support the objective.
- **Openness:** The meta-architecture is open to further extensions and architectural patterns as technologies mature and new patterns and components emerge. Technically, any IoT platform could be joined to the federation scope through

compatible and open interfaces. Implementation will base on Self-Sovereign Identities (SSI), that divide technical participation, trust anchors and business relationships into distinct entities.

- **Security:** Security is a cornerstone of any viable IoT implementation. Security measures can be implemented using different approaches, such as inter-DLT traceability. Regardless of technical implementation, security should be seen as of paramount importance.
- **Monetization:** The concept of data sovereignty builds improved tools to control security boundaries and privacy policies. In combination with regulations and legal constraints, the meta-architecture should embrace new business models, for example in smart contracts and contractual data sharing.

Technology objectives of the meta-architecture include defining a scalable, secure, open, federated and decentralised IoT meta-architecture that can be applied in varying use cases, including sensing, actuation, and smart behaviour of intelligent devices. Support for monetization and new business models is also integral part of the IoT-NGIN project. The meta-architecture supports domain horizontals providing actual IoT-NGIN use cases realizing actual architectural patterns.

The meta-architecture artifact is designed around four key artifacts: IoT Architectural Pattern Vertical, Domain Horizontal, Quality Vertical, and Element View. Each of the artifacts can be customised according to the architectural design needs and context. The elements of the meta-architecture are selected based on state-of-the-art prior work in the IoT domain. The meta-architecture provides a foundational framework for further development and there is room for evolution as technologies and implementations mature. For example, the quality attributes represent a large variety of possible non-functional quality requirements at the highest level. The quality features do not presumably need a lot of customization, rather the framing gives room for application-specific details.

Notably, in the meta-architecture view depicted in Figure 11, the Elements view is vertically supported by ML-powered AI technologies. This is of particular importance to the meta-architecture; with the number of interfacing platforms, devices and computational environments (e.g. near- and far-edge, fog, cloud) constantly rising, the next generation of IoT platforms and services is expected to arrange means for the operation of intelligent services based on ML capabilities. To this end, we consider that almost every composing element of the meta-architecture should be provided with the ability to make use of relevant AI services. In this context, ML as a Service operations are expected to emerge as necessary abstractions of such a modality, allowing for distinctive ML-related technologies to operate transparently in the physical-digital continuum on one hand (e.g. abstracting the link between physical and digital twins) and on the edge-fog-cloud continuum on the other.

D1.2 - IoT meta-architecture, components, and benchmarking

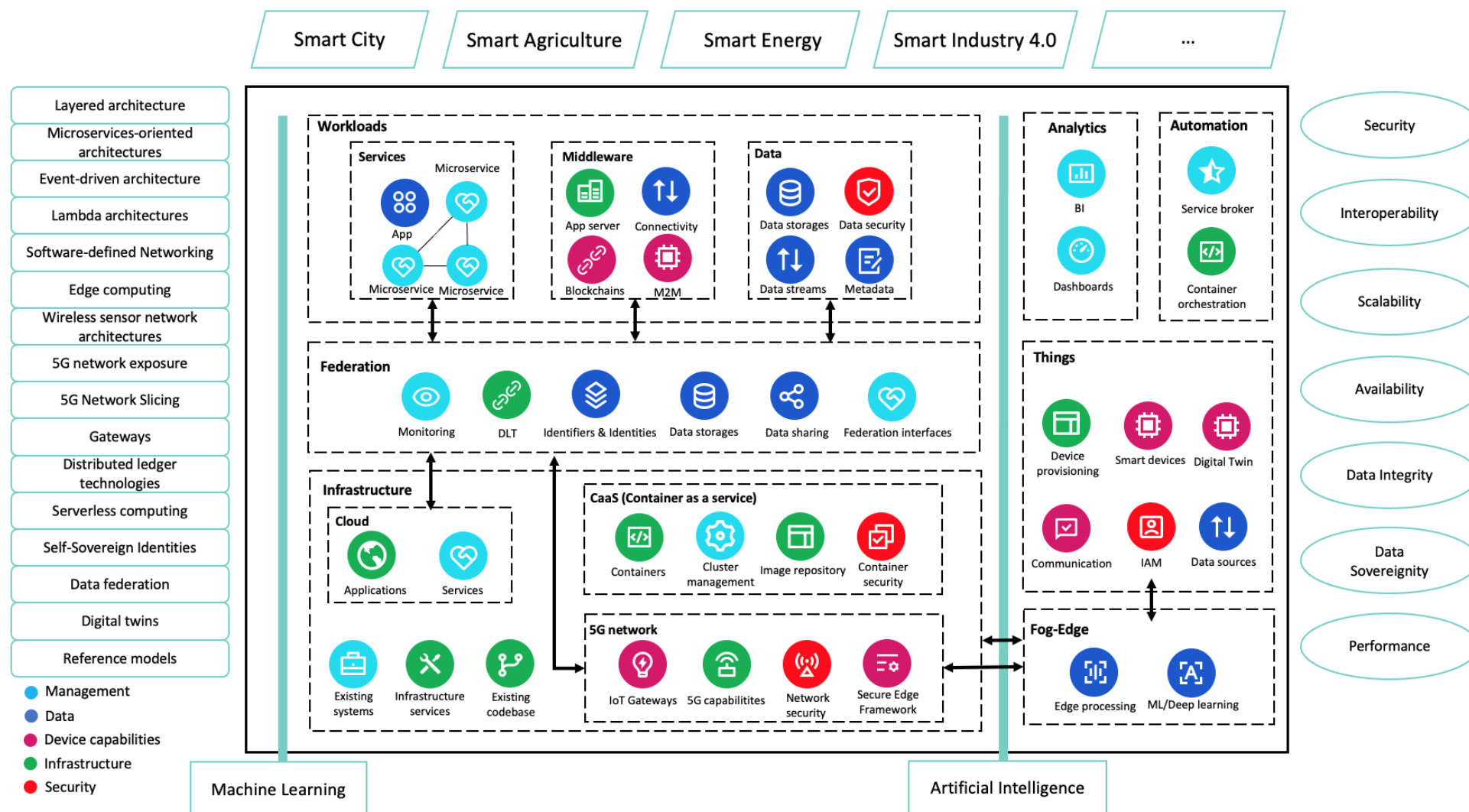


Figure 11: IoT-NGIN Meta-architecture.

4.3 Element view




The Element View provides a general technological view to the meta-architecture, with a focus on functional groups and components. Indeed, the functional groups are used to aggregate the functionality of the related components and, implicitly, provide an indication related to the objective of the group. The various elements have been further grouped with the use of labels, depending on the element's context. As also depicted in Figure 11, five distinct labels have been, thus far, identified, tailored to different element functionalities, namely Management, Data, Device capabilities, Infrastructure and Security. Note that the labels are color-coded to improve the readability of the meta-architecture and simplify its visualization. The approach of combining the functional grouping and component labels aims to bring the technology objectives to a more tangible level, also directly linking the technological or implementation design decisions closer to the quality attributes. Indicatively, security is important at every level of architecture but in IoT-NGIN there is special consideration in network security and secure edge framework. Similarly, under a data perspective, security is highly reflected in data sovereignty and integrity.




In the context of the meta-architecture and with reference to Figure 11, the following sections detail the identified functional groups and their composing elements. It is worth highlighting, again, that in all cases, it is assumed that AI services could (and should wherever possible) potentially be used to optimise the operation of all the functional groups, subgroups and elements presented below, as well as of the IoT platform-hosted application services. As the optimisation could be considered as an ubiquitous concern, spanning from 5G resources optimisation to predictive maintenance of the supporting hardware; the importance of ubiquitous AI is pertinent to almost all IoT platform scopes.

4.3.1 Things functional group

The Things functional group contains the elements related to the management, orchestration and proper application support of the far-edge IoT devices. In general, the latter may comprise a vast variety of devices, spanning from tiny sensors or semi-autonomous gateways to on-premises private infrastructures. Table 19 tabulates the elements that have already been identified as parts of the meta-architecture.

Table 19: Elements of the Things functional group.



Icon	Element	Label	Description
	Device provisioning	Infrastructure	The platform should be able to bootstrap the devices that are going to be interfacing with it.
	Smart devices	Device capabilities	The platform should be able to properly identify the capabilities of the interfacing IoT devices.
	Digital twins	Device capabilities	The platform should be able to support digital twins' functionality,

	Communications	Device capabilities	to enable high-availability and what-if scenarios. The platform should be able to support the various communication channels exposed and supported by the IoT devices
	IAM	Security	The platform should embed IAM functionalities to prevent unauthorised access and (re)use of resources.
	Data Sources	Data	The platform should be able to support a variety of data sources, depending on the hosted applications use cases.

4.3.2 Fog-Edge functional group

The Fog-Edge functional group refers to the ability of a platform to support, at edge or fog level, the execution of computationally (CPU- or GPU-wise) expensive applications on behalf of the platform-interfacing IoT devices. This typically implies the ability, at both device and platform level, to offload computing tasks from the devices to the edge/fog infrastructures in a secure and transparent way, then receiving and exploiting the processes results at device/far-edge level.

Table 20: Elements of the Fog-Edge functional group.

Icon	Element	Label	Description
	Edge processing	Data	The platform should be able to support edge processing (at compute level), lightening up the far-edge devices from the execution of heavy computational operations, via off-loading.
	ML / Deep learning	Data	Same as edge processing but related to supporting (possibly GPU-requiring) intensive machine learning operations (e.g. model training with data sovereignty considerations).



The ability to support this functional group in a secure, transparent, trusted and reversible manner will, eventually, deepen the integration among the devices, the edge, the fog and the cloud, leading to the emergence of a continuum of processes at the spatiotemporal domain, abstracting the IoT applications from their supporting infrastructure. Ideally, support for edge processing should be also accompanied by a simultaneous upgrade of the underlying wireless networking technology interconnecting the far-edge with the near-edge devices, ideally implying the existence of a slicing-enabled 5G network, towards reducing roundtrip communication delay and increasing the available bandwidth in order to be able

to accommodate various types of network traffic, depending on the use cases to be served by the IoT platform.

4.3.3 Analytics functional group

Arguably, the data offered by IoT infrastructures and services are widely considered to give rise to new business models and opportunities, particularly when combined with network edge processing [52]. The same holds for the data stemming from the platform operation; logs and error reports processing could give rise to predictive maintenance, sizing and resource management considerations that are essential for the healthy operation of a next-generation, large scale IoT platform. The analytics functional group is meant to support the ingested data refinement and transformation into valuable information insights through data analytics processes and assistive technologies. To assist the relevant data value unveiling, support for business intelligence operations and interactive dashboards are considered, as briefly presented in Table 21.

Table 21: Elements of the Analytics functional group.



Icon	Element	Label	Description
	Business Intelligence	Management	The platform should be able to offer controlled business intelligence services regarding the incoming data with respect to the core platform per se.
	Dashboards	Management	The platform should be able to offer graphical user interfaces in the form of dashboards to assist the data value discovery processes.

It should be noted that, in general, application-specific analytics services are expected to be served at application level (in the context of the Edge processing element of the Fog-Edge functional group as well as the Cloud and Container as a Service subgroups of the Infrastructure functional group described in 4.3.5) rather than at platform level.

4.3.4 Automation functional group

In the era of microservices, cloud native computing and 5G, it is essential to automate infrastructure and application provisioning, integration, and management. Indeed, considering the urge for rapid and scalable service deployments and granted the inherent management and platform integration requirements of the Edge-Fog functional group elements, the need to automate the platform operations and configuration is pertinent. The elements of the Automation functional group, briefed in Table 22, are related to enabling this automation perspective of a next-generation, edge-friendly IoT platform, catering on one hand on the proper operated services exposure and, on the other, on the management of the platform, per se, and of its hosted applications. In all cases, a container-like operation runtime is assumed, in line with the cloud native foundation definition [53], even though the container runtime could be relative to other, more edge-friendly technologies such as unikernels [54].

Table 22: Elements of the automation functional group.




Icon	Element	Label	Description
	Service Broker	Management	The platform should be able to provide simple integration of its services directly within the supported application platform.
	Container Orchestration	Infrastructure	The platform should expose management interfaces towards its easy setup, configuration and maintenance, but also towards simplifying the deployment of new applications and services.

The most well-known open source container orchestration framework at the time of writing this document is Kubernetes [55] which, at the same time, also offers service brokerage.

4.3.5 Infrastructure functional group

The infrastructure functional group contains elements related to the entirety of the infrastructure supporting the IoT platform bootstrapping, configuration, management and, in general, operation. An edge-oriented next generation IoT platform obviously needs to manage and build upon a variety of infrastructures including (usually virtualised) compute, storage, and network as well as other existing systems and infrastructures. Considering, further, the need for container-based operations (as per the Automation functional group) and the networking requirements implicitly imposed by the Fog-Edge functional group (essentially 5G with slicing support), the coordinated orchestration is as pertinent as ever.

Table 23: Elements of the Infrastructure functional group

Icon	Element	Label	Description
	Existing systems	Management	The platform should be able to integrate with existing systems and infrastructure components.
	Infrastructure Services	Infrastructure	The platform should be able to adopt new and manage existing infrastructure elements at the computing, network, and storage domains.
	Existing codebase	Infrastructure	Like the existing systems, the platform should be able to integrate with existing codebase.
N/A	Cloud	Subgroup	The platform should be able to interface not only with (far- or near-) edge, but also with cloud components and applications hosted on the cloud. The Cloud subgroup contains the relevant elements.
N/A	Container as a Service	Subgroup	Tightly interconnected with the Container Orchestration element of the Automation functional group, the Container as a Service



N/A	5G Networking	Subgroup	<p>subgroup allows for the API-based exposure of container-based IaaS services.</p> <p>The platform should be able to integrate and seamlessly support 5G capabilities such as 5G network slicing, Time Sensitive Networking and optimal 5G resource allocation.</p>
-----	---------------	----------	--

In the following sections, the elements of the Cloud, Container as a Service and 5G Networking subgroups are discussed.

4.3.6 Cloud subgroup

An Edge IoT platform is, by definition, meant to be interoperable with more powerful cloud infrastructures, the computationally (in terms of GPU or CPU) intensive tasks being offloaded to the cloud, if the accompanying latency requirements allow such an offloading process. Instead, in the cases where the latency requirements of a service are too strict, then the applications execution should stay at fog/edge level. Table 24 summarises the elements of the Infrastructure/Cloud subgroup that caters for such modalities.

Table 24: Elements of the Infrastructure/Cloud subgroup.





Icon	Element	Label	Description
	Applications	Infrastructure	The platform should be able to interconnect with cloud platforms at application level.
	Services	Management	The platform should be able to transparently manage their cloud parts, in the exact same way as the (near- or far-) edge ones.

In essence, the elements of the cloud subgroup imply the emergence of an edge-fog-cloud continuum, according to which, the applications should be able to be functionally operable regardless of the hosting execution infrastructure; in this way, load migration is rendered transparent, and the applications are granted with scalability and dynamicity (as to their performance) features.

4.3.7 Container as a Service subgroup

In tandem with the Container Orchestration element of the Automation functional group (see section 4.3.4), the Container as a Service subgroup elements ensure that effective infrastructure resources management and coordination is rendered possible, including Cluster management per se, Container operations (hosting and deploying containers), Container security and, of course, image repositories, as depicted in Table 25.

Table 25: Elements of the Infrastructure/Container as a Service subgroup.


Icon	Element	Label	Description
	Cluster Management	Management	The cluster supporting the platform should be manageable, in an easy, comprehensive, and predictable manner.
	Container security	Security	The cluster should be able to reduce the attack surface of the containerised applications to the minimum (at least under a firewall perspective) and ensure policy-based security of the containers.
	Containers	Infrastructure	The platform should be able to operate under a containerised (including unikernels and relevant lightweight resource abstraction technologies) manner, only.
	Image repositories	Infrastructure	The platform should have access to a set of image repositories so that the deployment of containers and their updates are possible.




It is worth mentioning that the Container as a Service subgroup comprises functionalities that are common in cloud-native environments as well. Indeed, among the most prominent projects graduated by the Cloud Native Computing Foundation [56] there exist several projects related to the aforementioned elements, such as Kubernetes (for container orchestration and cluster management), Containerd [57] (as a minimal Container runtime), OpenPolicyAgent [58] and Linkerd [59] (as policy based security and service mesh, respectively), TUF [60] (for managing container updates) etc, validating the need for such a subgroup.

4.3.8 5G networking subgroup

Networking lies at the core of every edge-oriented platform, interconnecting the physical world (devices, things) with the digital one (edge infrastructures). As real-time data is the driving force behind the vision of IoT, we need to make sure that this data is handled in a secure and trusted manner to be processed, while, at the same time, minimizing the end-to-end communication delays.

Table 26: Elements of the Infrastructure/5G networking subgroup.

Icon	Element	Label	Description
	5G capabilities	Infrastructure	The platform should have access to 5G networking capabilities, ideally accompanied with slicing and time-sensitive networking features, particularly to be able to satisfy the strict requirements of industrial IoT applications.




	IoT gateways	Device capabilities	To enable IoT communications, usually device-to-device or device-to-cloud, the platform should integrate IoT gateways with support for simple data filtering towards enabling visualization and complex analytics are required.
	Network Security	Security	The platform should be able to exploit the entire set of 5G network security features such as end to end encryption, active cyberattacks prevention (effectively mitigating security and privacy threats under 5G), etc.
	Secure Edge Framework	Device capabilities	In tandem with the Edge-Fog functionality group elements, at network level, the devices should have the ability, to consume such services.




In any case, the existence of the infrastructure/5G networking subgroup elements, reassure that the edge-centric vision of IoT-NGIN regarding next-generation IoT platforms is, actually, plausible.

4.3.9 Federation functional group

The Federation functional group, targets at ensuring scalability, availability, and stability of the next generation of IoT services as well as sovereignty and transparent control of data and data streams. Data access should be, on one hand, controlled but, on the other, data are exploited to unveil their maximum potential. The elements and subgroups of the Federation functional group ensure that access to services and data is offered to the end-users regardless of their location or the time-of-service request and that the quality of experience (QoE) they enjoy is acceptable at all times. Table 27, below, tabulates the core elements and subgroups of the Federation functional group.

Table 27: Elements of the Federation functional group

Icon	Element	Label	Description
	Monitoring	Management	The platform should provide enough tools and services enabling real-time active monitoring so that optimisation actions may be performed as soon as possible, and downtime/degraded performance periods are minimised.
	DLT	Infrastructure	To support data non-repudiation and support the ledger and inter-ledger processes, the platform should feature DLT support, e.g. in the form of blockchains-flavoured applications.
	Identifiers and Identities	Data	The platform should be able to either provide identity management services or catalyse the

	Data Storage services	Data	secure use of self-sovereign identities, at federation level. The platform should provide, at federation level, data storage services towards increasing efficiency, reliability, and performance.
	Data Sharing	Data	The platform should offer structured services towards data sharing in an open, transparent manner. The data sharing features should be available at federation level.
	Federation Interfaces	Management	The platform should support the federation by appropriate federation management interfaces.

It should be highlighted that the Federation functional group and subgroup elements are meant to orchestrate data/control flows, data streams, services, and infrastructure at federation level. However, at the same time, multiple levels of control contexts should be available, so that (data or identity) sovereignty is respected at all times, e.g. data sharing services should be available at federation level, however one should be able to apply more restricting sharing policies for accessing certain data contexts e.g. though the definition of edge-level access.

4.3.10 Workloads functional group

The term "Federation" implies platform consistency across the entirety of the operations of the platform at computational level, also known as *workloads*. Since data operations play a central role in next-generation IoT platforms, platform workload consistency spans not only the offered web services but, also, the handling of the data operations, in general, including data management lifecycle, in particular. Under that perspective and considering the diversity of devices and protocols formulating the current (and future) IoT landscape, protocol management towards effective *interoperability* is of paramount importance. The subgroups of the Workloads functional group that ensure the aforementioned "platform consistency" from a workload perspective are tabulated in Table 28.

Table 28: Elements of the Workloads functional group.

Icon	Element	Label	Description
N/A	Services	Subgroup	The platform should be able to operate and expose application services and manage their integrations at federation level.
N/A	Middleware	Subgroup	Should support multiple middleware-based adaptation services, so that different protocols and technologies may be supported, at federation level, boosting the platform interoperability.



N/A	Data	Subgroup	The platform should be able to securely manage and offer services related to data and metadata streams management, in various contexts and modalities.
-----	------	----------	--

As expected, due to their inherent federation-oriented nature, the various subgroups of the Workloads functional group directly interface with the Federation functional group elements, in an attempt to achieve graceful, orchestrated coordination of the offered services. In the following sections, the elements of the above subgroups are briefly described.

4.3.11 Services subgroup

The first consideration of a workload federation are the exposed services, building up the core of the IoT platform computational environment, including the core platform services and the hosted (user-oriented) ones. Apart from the services operations, the integration potential scope is crucial so that an ecosystem around the IoT platform can boom. Table 29 highlights the elements of the Services subgroup.





Table 29: Elements of the Workloads/Services subgroup.

Icon	Element	Label	Description
	Apps	Data	The platform should be able to host, operate and expose the functionality of applications and services in a transparent manner, catering for their performance optimisation.
	Microservices	Management	The platform should be able to operate on top of and support microservices-oriented architectures.

4.3.12 Middleware subgroup

In tandem with the data-oriented character of the Federation/Workloads subgroup and the Integrations element of the Federation/Workloads/Services, the ability to interact with the IoT environment in a variety of ways (implying the use of different protocols) is critical. The Workloads/Middleware subgroup elements focus on the interoperability potential of the platform, considering not only the classical client-server, HTTP-based protocols but, also, other technologies, friendlier to IoT scopes relevant to self-organization and data security and sovereignty. Table 30 summarises the elements of the subgroup.



Table 30: Elements of the Workloads/Middleware subgroup



Icon	Element	Label	Description
	Application Servers	Infrastructure	The platform should be able to run application servers hosting arbitrary technologies and protocols.
	Blockchains	Device Capabilities	To support the DLT (Blockchains) services potentially offered by the platform (e.g. for SSI or to ensure data non-repudiation in mission-critical operations), the devices interfacing with the platform should feature hardware and software able to support (preferably hardware accelerated) blockchain operations.
	Connectivity	Data	In order for the IoT devices to interface with the platform and vice-versa, compatible communication capabilities should be featured in both ends. Ideally, this element should be linked with 5G capabilities, hinting towards the Fog-Edge functional group and the infrastructure/5G Networking functional group.
	M2M Services	Device Capabilities	Since the interfacing among the devices and the edge/cloud nodes should be automated, machine-to-machine communications should be, mutually, supported.

4.3.13 Data subgroup

Having considered the communications and services part of the federation workloads, the final step towards achieving operation is to, actually, enable low-level data operations, per se, both from an infrastructural point of view but also from the data and metadata management perspective. The Workloads/Data subgroup elements, briefly described in Table 31, outline the relevant functionalities that should be considered.

Table 31: Elements of the Workloads/Data subgroup.

Icon	Element	Label	Description
	Data Storage	Data	The platform should expose federated data storage services, ensuring that the data is always available, in the modality required (e.g. block storage services, object storage services).
	Data Security	Security	The platform should be able to assure that the data is secured throughout their entire lifetime, including transmission, manipulation, storage and, in general, access.

 	Data Streams	Data	<p>The platform should be able to cope not only with batches of data but, also, with a variety of data streaming technologies, to support the vision of real-time computing.</p>
	Metadata management	Data	<p>The platform should be able to offer services related not only to data but, also, to metadata, so that data preselection, selection and processing is accelerated, and feature extraction is facilitated.</p>

4.4 Architectural Patterns Vertical

The Architectural Patterns Vertical part of the IoT-NGIN meta-architecture summarises several architectural viewpoints that have been identified and can be applied, at design level, to drive the deployment of effective, efficient, adaptive, performant, secure and private IoT platform services and applications. The identified architectural patterns cover all the possible aspects of a next-generation IoT platform, effectively leading to the introduction of several of the functional groups, subgroups, and elements of the meta-architecture's Element view. Further, all the identified architectural patterns contribute, in many ways to the satisfaction of the Quality Vertical elements, documented in section 4.5.

The already identified architectural patterns are shown in Figure 11 and have been detailed in section 3.

4.5 Quality Vertical

The quality attribute vertical illustrates the generic functional and non-functional requirements of the IoT-NGIN meta-architecture. These attributes can be prioritised through different means and in function to the selected combination of patterns and use cases. The Quality Vertical is of great significance since it summarises architectural design needs that are common to many of the use cases.

4.5.1 Security

The essence of IoT is in the connectivity and data transmission between a diverse set of devices and services. While the IoT field helps to build a smart and digital world, it means that the technology is exposed to many adversaries. The core security themes of IoT are [61]:

- **Confidentiality.** Confidentiality guarantees that only authorised entities are allowed to view, edit and use data throughout its life cycle.
- **Integrity.** The data is not tampered with by any non-authorised entity. Integrity is critical in IoT because tampered or malicious data can affect operational activities and cause large-scale disruption or danger to end-users. Mechanisms such as access control and false data filtering schemes aid in ensuring data integrity.
- **Availability.** IoT data, devices and services should be available when requested. Depending on the application, high availability of "five nines", or 99.999 percent

could be required, translating to 5.26 minutes of downtime per year. Many applications depend on real-time communication or data processing, and if the real-time principle is under threat, the effect can be severe. Denial of service (DoS) attack is a general method used by adversaries.

- **Identification and Authentication.** By using identification, unauthorised thing entities cannot connect to IoT devices or networks. Authentication then ensures the validity of transmitted data and the validity of the client or device requests. Many open-source and commercial IoT platforms offer embedded IAM tools.
- **Privacy.** Only the owner of the data can control the data or services, other entities cannot access or process the related information. Privacy is important in IoT since many users share the same devices, services and general infrastructure.
- **Trust.** Trust is a characteristic that bases on the other security features. The trust takes place between the IoT devices, between devices and applications, and between end-users and services. IoT-NGIN can build trust in its technology platforms by adapting inter-DLT technologies and applying industry standardisation in the federation infrastructure.

4.5.2 Interoperability

Interoperability is one of the most vital topics across the IoT field. Traditionally interoperability has been defined as a ability of two or more systems or components to exchange information and to use the information that has been exchanged. As the concept is broad, interoperability can be categorised into several levels: device, network, syntactic, semantic, cross-platform, and cross-domain levels. The architectural patterns partly address the interoperability problem space. For device level, gateways and communication protocols solve connectivity challenges. SDN and edge processing focus on the network level. Technologies such as REST, semantic web services, or WoT attempt to build enhances syntactic and semantic interoperability [62]. Cross-platform and cross-domain interoperability can be achieved with combination of methods, such as open interfaces, inter-DLTs, and ultimately adopting governance guidelines using, for example IDS.

4.5.3 Data sovereignty

Traditionally sovereignty is linked to authoritative and institutional claims, and thus data sovereignty frequently addresses different kinds of data flows through national regulations and laws. Data sovereignty is perceived as a concept where individuals, organizations, communities, and other groups should be able to have control over their data.

Data sovereignty is a complex and essential requirement in the next generation of the Internet and IoT. The complexity arises from the fact that data sovereignty can be applied to any applications that transport data across geographic locations. It includes concepts such as the role of national governments and their influence on data that is stored in domestic or foreign clouds, geographic and Indigenous data sovereignty, and patient health data sovereignty.

4.5.4 Scalability

Applications in the IoT domain should scale up horizontally and vertically. Horizontal scalability is adding the number of devices to a network and vertical scalability increases the performance of a single device or application. Infrastructure must also allow a growing number of user requests and actions. More complex data ingestion, transmission, processing and analysis is causing heavy pressure for solutions in data storage, communications, and analytics. Furthermore, scalability in IoT is restricted by factors such as lack of open and coherent standardization, protocols, and device and service discovery [63]. The ultimate goal of IoT-NGIN is to provide means for full horizontal scalability with stateless applications and full vertical scalability at the cloud level.

Federation of ML and deep learning models and performing training operations locally and on the network edge can considerably increase the scalability of applications related to predictive and behavioural analytics.

Utilizing Inter-DLT instead of a single DLT or blockchain can also increase the scalability of DLT applications. In addition, the new federation models might have an indirect scalability impact on use cases unrelated to DLTs. The reasoning is that through the openness of the IoT platforms, more collaborators can participate in the ecosystem, which accelerates adaption rates for certain technologies, for example, more efficient data communication protocols.

4.5.5 Flexibility

Flexibility is a quality attribute that can be linked to many other attributes. In IoT-NGIN context, flexibility has close relations with semantic interoperability, performance and manageability. IoT applications should withstand changing environments, for example, technologies, interfaces, constraints and development methods, among others.

The meta-architecture artifact contains innate flexibility since the building blocks are not locked into static sets of use cases or architectural patterns. IoT-NGIN creates a DLT-enabled meta-level Digital Twin (MLDT) that allows on-usage interpretation and application of data and information.

4.5.6 Performance

Performance is a complex topic in IoT because the field consists of many different technologies. Performance evaluation can target devices, networks, protocols, middleware, platforms or more specific building blocks such as service integration.

Other common IoT data processing characteristics are message parsing, statistical and predictive analysis. IO operations are also a significant part of IoT data flow, and many applications need external access to data storage or messaging services. Frequently used performance metrics for Distributed Stream Processing Systems (DSPS), such as Storm, Flink or Spark, are: latency, throughput, jitter and CPU/memory utilization. Similar metrics are applicable to IoT-NGIN since many use cases are related to real-time stream data processing [64].

4.5.7 Reliability

Reliability is a system design challenge arising from various factors. In IoT, the unpredictability of the environment is one of the biggest factors, alongside inconsistent end-user input and usage. Key reliability requirements are the heterogeneity of the IoT nodes and network, in parallel with resource-awareness and environment dynamism.

The reliability requirement must address errors in hardware, software, and data. Especially the data component regularly contains detection, transmission and analysis errors that lead to false information. One additional dimension is that the severeness of the errors differs in magnitude. Some errors are minor, while other faults can lead to system failures or poor user experience. Open challenges are to efficiently address both correlated and unpredictable failures.

IoT-NGIN focuses on network reliability and provides standalone 5G networks for massive Machine Type Communications (mMTC). In mMTC data and control signals are re-transmitted to improve reliability in packet reception.

4.5.8 Manageability

Manageability in IoT is commonly related to device management challenges. From the meta-architecture viewpoint, the manageability aspect highlights only general and technology-agnostic approaches. On the application level, implications and direction for certain technologies need to be considered on individual bases. The manageability challenge is also linked to the domain horizontal, where use cases differ on the scope and management needs.

4.6 ML and AI layer

ML and AI verticals in IoT-NGIN meta-architecture (Figure 11), conforms to the Artificial Intelligence and Big Data services provided by the project. Machine Learning as a Service (MLaaS) platform will enable the application of AI in different use cases, living labs, and any IoT platform planning on leveraging Machine Learning and/or Big Data technologies on its use cases. This MLaaS platform will run on Edge Cloud nodes and it will be able to deploy ML models on IoT Devices seamlessly as presented in the left hand side of Figure 11.

The presence of AI in the IoT-NGIN meta-architecture is ubiquitous and its functionality may be shared among the rest of the components in the meta-architecture. Even though the execution of ML tasks is bounded to fog-edge nodes, the infrastructure shall be able to escalate the operations to (i) the 5G network as part of the Secure-Edge framework, to (ii) the cloud as an application, or (iii) as a container managed by the CaaS (Container as a Service), depending on the requirements of the ML tasks and the target environment. Thanks to this infrastructure interoperability, the ML tasks can be connected to other components of the meta-architecture. This means that the ML operations will be able to access the data storage and the data streams that are federated, as well as to access the data sources themselves. It is important to mention that these ML operations will be supported by Edge processing capabilities when they are available. In this regard, the infrastructure shall be in charge of giving access to processing power to the ML tasks at any time, leveraging on fog-edge nodes the execution when possible.

D1.2 - IoT meta-architecture, components, and benchmarking

IoT-NGIN has already made progress on the MLaaS platform, which are described in Deliverable D3.1 [65]. The first version of the architecture for this platform is presented in Figure 12. The MLaaS architecture design is compliant with IoT-NGIN's meta-architecture. First, the MLaaS edge nodes, which represent the fog-edge nodes in the meta-architecture, are where most ML-related operations will take place. What's more, the edge nodes of the MLaaS are designed to work in a federated manner, as described in the meta-architecture, supporting the development of a federated ML framework. In addition, even though it is not represented in the figure, the MLaaS edge nodes operations are supposed to be either enclosed within the Secure Edge Framework of the 5G network. An another option is to utilise a container managed by the "Container as a Service" component of the infrastructure in the meta-architecture. Secondly, the edge nodes are directly connected to the IoT Devices for (i) deploying ML models and (ii) accessing data sources directly, a representation similar to the "fog-edge nodes" connection to the "Things" component in the meta-architecture. Thirdly, the MLaaS architecture may access cloud resources of the network, just as represented in the "Infrastructure" component of the meta-architecture. Further information regarding details, actors, platform use cases, requirements, logical view and data management procedures, in UML shape, is collected in D3.1.

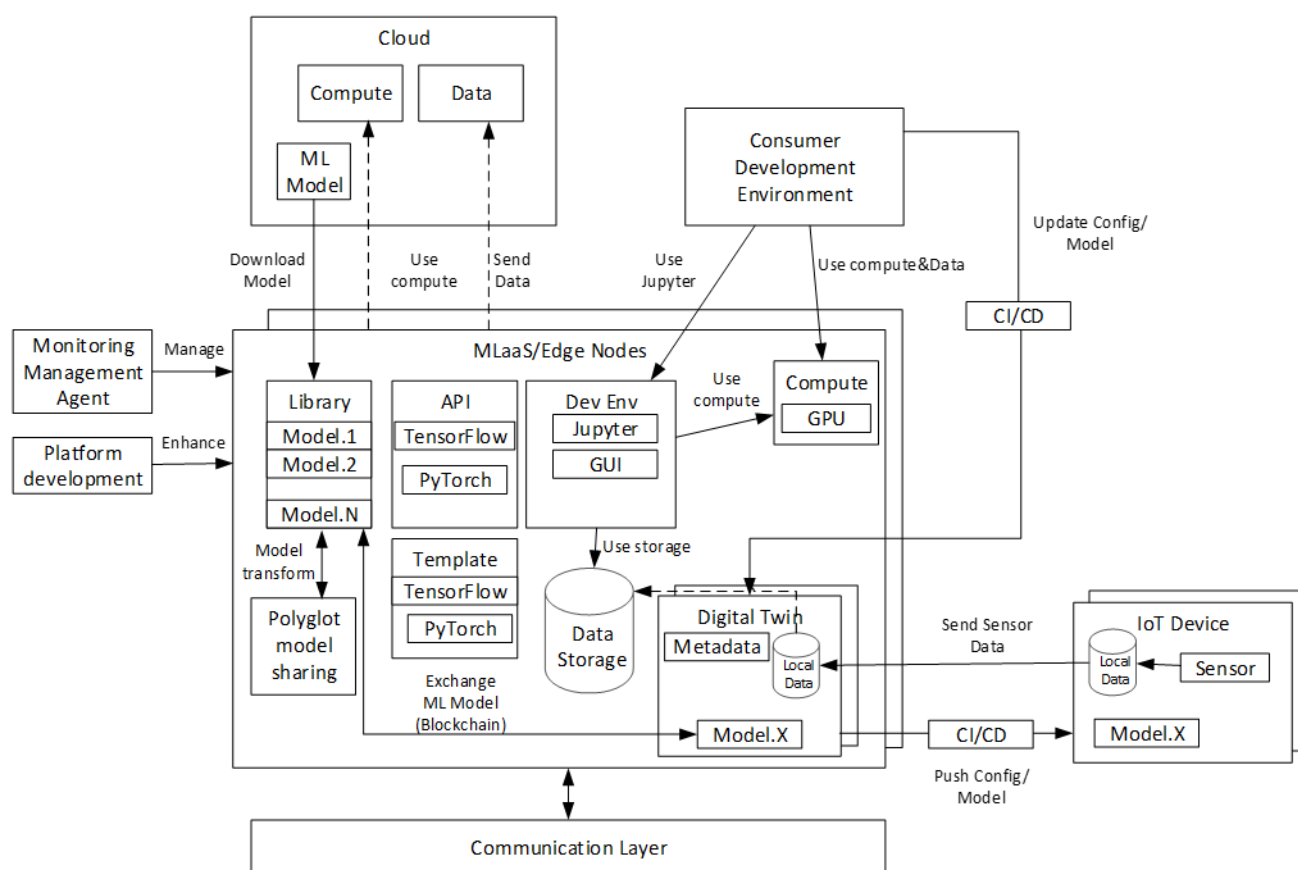


Figure 12: MLaaS Architecture, first version

All in all, the MLaaS architecture design not only is compliant with the meta-architecture, but it also provides an innovative model of the next generation of IoT Intelligence, where Artificial Intelligence capabilities are federated and distributed among the edge nodes, providing new means for the application of machine learning models across different environments.

4.7 Compliance to the IoT-NGIN meta-architecture

The inherently generic nature of the IoT-NGIN meta-architecture allows modelling practically all IoT-relevant architectures, in a consistent manner. Under a more contextualised perspective, though, the meta-architecture should, eventually, guide the design of next-generation IoT platforms, networks and applications, also hinting towards the steps required to transform a “traditional” IoT platform approach into a next-generation one.

Indeed, several of the IoT-NGIN meta-architecture notions and technology spaces (mostly in the form of elements), as presented above, are drawn from existing, traditional IoT platforms. In IoT-NGIN, we argue that, to ensure compliance with the identified meta-architecture principles, elements from all three different meta-architecture views should be adopted, though the level of adherence differs among the various views.

Table 32 gives an overview of the compliance matrix of the IoT-NGIN meta-architecture and which elements of the meta-architecture should be adopted by an IoT platform so that it may be considered compatible with the meta-architecture.

Table 32: IoT-NGIN meta-architecture compliance matrix.

Meta-Architecture view	Compliance baseline
Quality vertical	ALL elements/requirements should be satisfied.
Architectural patterns vertical	At least one should be adopted ¹ .
Element View	All security elements should be present. For each one of the first-level functional groups, at least one element should be adopted.
Underlying ML/AI	Desirable, not necessary.

As per Table 32, all key elements should be adopted at design level from the Quality vertical (see section 4.5). Indeed, the Quality vertical dictates the principal values that every next-generation IoT architecture should build on, so that is relatively under both a technical and a business perspective may be claimed. Elements such as security and data sovereignty are always pertinent, particularly in the era of GDPR, and should not be overlooked. Similarly, scalability, flexibility and performance considerations should be considered to guarantee viability at business level, in tandem with interoperability, reliability and manageability.

Dissimilar to the Quality vertical case, the Architectural patterns vertical comprises several design patterns and notions, each one introduced to address different IoT designs and specifications. Granted the heterogeneity of the IoT use cases and the diverse landscape of the application requirements that would need to be served by a framework compliant to the IoT-NGIN meta-architecture principles, it is not possible to distinguish a concrete set of architectural patterns that should be adopted. However, at least one should be employed,

¹ With the exception of the identification of new patterns that have not been (implicitly or explicitly) considered by the relevant work. See below for details and discussion.

to ensure that the design of the IoT platform at hand relies on existing, verified patterns that have been previously employed. Notably, granted that the list of patterns is not exhaustive and more will be added as a result of the *Technology watch* activities of the project, it is possible that another pattern (e.g. a cloud-oriented design pattern like the Ambassador [66] or the Choreography [67]) is considered by the IoT platform at hand. In such a case, omitting the already identified patterns in favour of a non-acknowledged one is considered acceptable.

Finally, with respect to the Elements view, at least all first-level functional groups should be adopted, namely at least one element from each of the functional groups should be employed, the security-related elements being necessary. Indeed, the IoT-NGIN meta-architecture first-level functional groups and their necessity is tabulated below.

Table 33: The meta-architecture functional groups

Functional Group	Necessity
Infrastructure	Infrastructure provisioning, bootstrapping and management is necessary for allowing applications to properly operate.
Federation	Granted the (horizontal) scalability requirement stemming from the Quality vertical, support for federated orchestration at compute and data level is deemed necessary.
Workloads	In tandem with the Infrastructure and the Federation functional groups, the workloads functional group represents the necessity to optimally manage the hosted application and data workloads.
Things	The Things functional group functionalities cover the necessity to manage not only the platform-oriented workloads, but also the device ones.
Fog-Edge	The Fog-Edge functional group is necessary to achieve effective communication and secure task offloading (towards lowering the cost and increasing the capabilities of the supported IoT devices)
Automation	Granted the expected rise in the design complexity of the offered IoT platform services, the Automation functional group represents the needs to easily install and dynamically (automatically) manage the provided platform instances.
Analytics	To uncover the value of the underlying platform-collected/generated data, it is important to feature analytics toolboxes.

Regarding the underlying ML/AI layers serving not only the platform functionalities but also the potentially hosted applications, their employment is desirable but not entirely necessary. However, the vision of IoT-NGIN regarding next-generation IoT platforms accompanied by self-optimizing functionalities clearly hints towards their global adoption.

5 The IoT-NGIN architecture

The IoT-NGIN architecture follows the path laid by the meta-architecture, exhibiting an extensible, modular design that allows the decoupled deployment of functionalities and services under a semi-orchestrated manner. Moreover, its design allows provisioning either in the form of Software as a Service (SaaS) or Platform as a Service (PaaS). Figure 13, below, depicts the IoT-NGIN architecture.

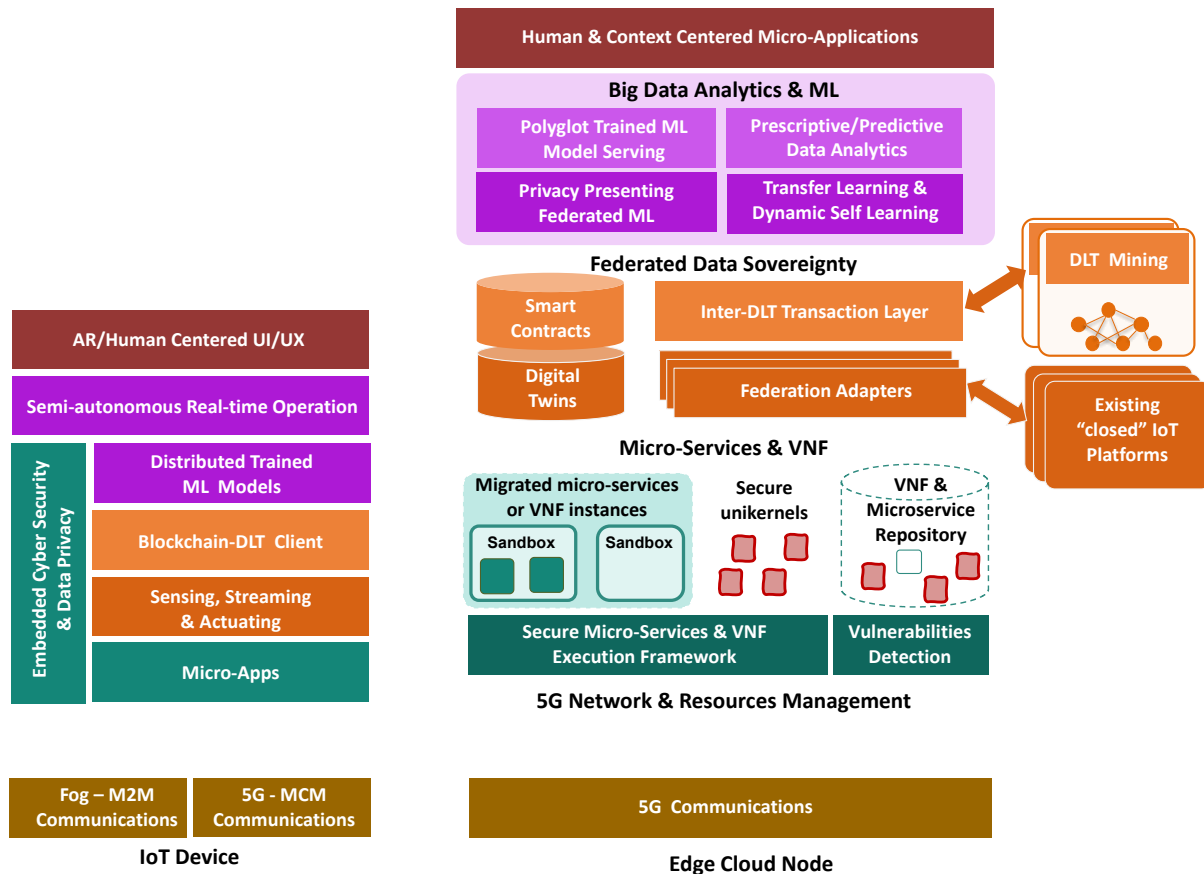


Figure 13: The high-level architecture of IOT-NGIN.

As shown in Figure 13, the IoT-NGIN high-level architecture may be split into the following groups of components:

- Federated Communications (brown colour):** realisation of communication and resources management functionalities, consisting of a set of IoT/5G optimisation functions (e.g. 5G relays optimisation, slicing, integration of TSN components), M2M and Further Enhancement to Device-to-Device (FeD2D) enhancement by reducing CCI between devices which share the same uplink, along with network resource management and self-aware IoT resources. Instead of a single communication technology, independent and different communication protocols may be used in parallel and, where it is possible, to replace one technology with another, offering dynamic network connectivity. IoT-NGIN integrates virtual networking management tools (e.g. Management and Orchestration – MANO – frameworks aligned with the

ETSI NVF specifications [68]) to directly manage dedicated network slices, with specific characteristics to meet highly demanding digital services requirements.

- **Micro-services and VNFs (turquoise colour):** a “living” collection of functionalities, implemented as container-based micro-services or unikernels. IoT-NGIN works on a novel Secure Micro-Services Execution Framework implementing operational tasks in close interaction with the micro-services. The framework comes with native thread-safety considerations, its implementation being ongoing using the RUST programming language [69]. This unique framework will enable migration and hosting of both unknown/potentially malicious and fully trusted/digitally signed micro-services instances. Of course, the management of the relevant micro-services and VNFs is laid upon the responsibility of the adopted MANO framework, using Kubernetes as the virtual infrastructure manager (VIM) of choice. This approach allows IoT-NGIN to operate using the latest standards at the level of network management, coupled with state-of-the-art, cloud-native technologies with respect to data and container orchestration. Particularly when it comes to data management, the project has already considered the employment of cloud-native data storage solutions (via the combination of the rook [70] and ceph [71] frameworks) to ensure reliability, replicability and scalability as to the data lifetime management.
- **Federated Data Sovereignty (orange colour):** To make the IoT-NGIN architecture as scalable as possible, IoT-NGIN adopts some of the emerging Decentralised Identifier (DID) and Self-Sovereign Identity (SSI) technologies and applies them to enhance data sovereignty and privacy. The security of data sharing federation will be based on DLTs. However, instead of using a single DLT, the IoT-NGIN federation uses an inter-DLT approach, where multiple independent and different distributed ledgers may be used in parallel and where it is possible to gradually replace one ledger technology with another, if needed. At the same time, by exploiting the SSI technologies (either coupled with DLTs or not), IoT-NGIN allows effective identity management without depending on external identity management systems, essentially giving the end-users with control not only of their data, but also of their identity. Interestingly, within the context of IoT-NGIN, the SSIs are co-designed together with the meta-level Digital Twins, so that on one hand the control of the identity of the physical twins does not change hands, at the same time guaranteeing (virtual) device identity uniqueness at federated level.
- **Federation of Big Data Analytics & ML (purple colour):** In the context of IoT-NGIN, a ML as a Service (MLaaS) framework is being designed and developed, offering ML/AI services to the entirety of the IoT-NGIN platform components, but also the hosted applications. The IoT-NGIN MLaaS framework is based on state-of-the-art open-source frameworks (Kubeflow [72]), essentially implementing common ML operations such as pre-processing, training, model serving and inference, on demand. The ML pipelines are employing a continuous integrated approach towards the emergence of “live” models, even in batch learning cases. Beyond the state of the art semi-supervised and unsupervised deep learning/reinforcement learning techniques and privacy preserving federated ML transfer learning are implemented to train ML models while keeping the IoT data in their original locations, and unsupervised cross-context transfer learning (e.g. cross-context AI by design), with inline adaptive self-learning is studied in order to improve the resulting machine learning models. Moreover, ML- and zero-knowledge proof-based cybersecurity components based on Generative Adversarial Networks (GANs, [73]) are developed to identify malicious IoT nodes and data poisoning attacks, as well as timely identify intrusion detection [74].

- **Human-Centred Augmented Reality Tactile IoT (dark red colour):** IoT-NGIN has a special focus on the enhancement of human-centred IoT devices discovery, recognition, and novel ambient intelligence-based control, combining physical access, user rights/groups and IoT ownership. To this end, IoT NGIN strives towards the generation and maintenance of a repository sharing IoT-AR enhanced UX/UI SW components, tools and libraries to 3rd Parties.

When it comes to compliance with the meta-architecture, the adopted architecture re-uses many of the architectural patterns, as tabulated below.

Table 34: Compliance of the IoT-NGIN architecture with the meta-architecture; architectural patterns.

Architectural Pattern	IoT-NGIN architecture consideration
Microservices-oriented architecture	IoT-NGIN architecture foresees a cloud-edge execution framework supporting container- and unikernel-based loads in a secure and trusted manner. The microservices are adopting the emerging paradigm of CaaS (Container-as-a-Service)
Event-driven architecture	The IoT-NGIN MLaaS framework employs event-driven architectures and technologies (such as Apache Kafka [75]) to orchestrate in real-time the various ML pipelines and management operations.
SDN	IoT-NGIN relies on the use of container- and unikernel-based technologies, orchestrating them with the help of Kubernetes. The internal communications of the various components are heavily relying on SDN, as do the network interfaces exposed by the 5G network slicing framework developed by IoT-NGIN.
5G network exposure	In the context of IoT-NGIN, the following 5G resources will be offered for experimenting in the laboratory: network and operational management capabilities and cloud infrastructure resources.
5G network slicing	IoT-NGIN works on the emergence of an open 5G network slicing framework able to dynamically manage the end-user requirements in terms of performance or reliability (at network or service level) and match them with the available resources at operator level.
DLTs	<p>From IoT-NGIN point of view, DLTs are important for many reasons. IoT-NGIN has a vision for systems open to everyone. DLTs allow parties that do not fully trust each other to collaborate in a single system without requiring a (costly) trusted third party to facilitate the cooperation. DLTs also provide a high degree of automation through smart contracts and similar technologies.</p> <p>Self-Sovereign Identities (SSI) provide a technology to separate technical participants from other trust elements and relationships.</p>

SSI	Security by design is fulfilled with inter-DLT traceability and demonstrated with a DLT-based meta-level digital twin (MLDT). SSI is used as an identifier solution to improve privacy, security, automation, and to give users more control of their identifiers. In particular, they are used as identifiers and credentials for the Digital Twins.
Data federation	IoT-NGIN relies on state-of-the-art technologies and frameworks to ensure consistent data availability at the spatiotemporal domain, also catering for data storage reliability employing virtual storage replicaset and associated infrastructure allocation rules.
Digital twins	In the context of IoT-NGIN, digital twins are employed to facilitate interaction of the platform with their physical twins (and vice versa), increase service reliability and boost the performance of data services provisioning. Coupled with DLTs and SSIs, meta-level Digital twins are employed to increase lifetime trackability and traceability.

Similarly, the IoT-NGIN architecture is compliant to the Quality Vertical requirements, as per Table 35.

Table 35: Compliance of the IoT-NGIN architecture with the meta-architecture; Quality Vertical.

Requirement	IoT-NGIN architecture consideration
Security	In the framework of its security considerations, IoT-NGIN employs beyond the state-of-the-art approaches (GANs) to identify intrusion detection at network level and, also, to identify fraudulent behaviours at the level of local ML models poisoning. To further fortify the next-generation IoT platforms, IoT-NGIN employs ML-powered "perimeter defence" in the form of dynamic honeypots deployment and auto-configuration, in tandem with a network-level vulnerability crawler targeting at uncovering possibly vulnerable IoT nodes or services, then feeding the dynamic honeypots deployment service to trap the attack agents into attacking the wrong IoT nodes/services and exposing themselves.
Interoperability	Interoperability is one of the key considerations of IoT-NGIN technology design when it comes to interfaces specification and data models selection. Section 6 presents an indicative set of data models and interfaces that have been considered at project level.
Data sovereignty	IoT-NGIN employs DLTs and SSIs to ensure that end-users have control of their data. By means of the inter-DLT technologies adopted and advanced in the context of the project, end-users-

	based, dynamic rules for data sharing at local or global level will be rendered possible.
Scalability	The design of IoT-NGIN heavily relies on cloud-native principles, deeply adopting a carefully designed stateless, micro-services-oriented architecture governed by state-of-the-art, open-source container-orchestration technologies (Kubernetes). This allows IoT-NGIN to scale not only horizontally but, also, vertically, being able to consume more resources when the computational load is high and less resources, otherwise.
Flexibility	The IoT-NGIN architecture is flexible since practically all IoT-NGIN components may be removed and substituted with others, if needed. Interoperability at interface and data model level should ensure that integration of the newly introduced components is seamless.
Performance	As per the scalability requirement, at compute level, IoT-NGIN is able to horizontally scale and make use of all the available computational resources. At the same time, the IoT-NGIN architecture prioritises 5G when it comes to networking; combined with network slicing, the architectural considerations regarding network performance are state-of-the-art. Finally, when it comes to the MLaaS IoT-NGIN framework, GPU acceleration is natively supported at both design and implementation level.
Reliability	The technology stack on which the IoT-NGIN architecture is built allows for the determination of replicaset (with configurable minimum, normal and maximum instance number) at the level of both service provisioning and data storage (at both block and object level). Combined with the high-availability features of 5G networks in the context of slicing, we consider that the reliability requirement is, by design, satisfied.
Manageability	The IoT-NGIN architecture and technology basis heavily depends on open-source, open-API-based frameworks, known for their manageability (e.g. [76], [77]). The same goes for the direction of the project's 5G network exposure API research as per §3.8.

5.1 Mapping the IoT-NGIN architecture to the meta-architecture

Finally, when it comes to compliance with the Elements View, Table 36 highlights the relevant considerations at architectural level.

Table 36: Compliance of the IoT-NGIN architecture with the meta-architecture; Elements View.

Functional Group	IoT-NGIN architecture consideration
Infrastructure	<p>The IoT-NGIN architecture is flexible and can accommodate existing platforms and services and exhibits cloud-native support at technology level to host and optimally manage and services applications. Supported elements:</p> <ul style="list-style-type: none"> • Existing systems • Infrastructure services • Existing codebase • Cloud/Applications • Cloud/Services
Federation	<p>The IoT-NGIN architecture treats DLTs as first-class citizens when it comes to data management and sovereignty. Further, by supporting SSI, it fully complies to the Identifiers and Identities element. The use of the cloud-native technology stack for the compute and data management (Kubernetes, rook and ceph), satisfies the requirements posed by the Monitoring (with the help of Prometheus [78]), Data storages, Data sharing and Federation interfaces elements. Elements supported:</p> <ul style="list-style-type: none"> • Monitoring • DLT • Identifiers and Identities • Data storages • Data sharing • Federation interfaces
Workloads	<p>At service level, IoT-NGIN relies on a microservices-oriented architecture to offer its services. Further, several DLT (blockchains) protocols are considered (particularly in the context of inter-DLT-based communications). Further, at the level of federated communications, M2M services are considered. Last, at data level, the architectural support for the emergence of a data federation managing the entire lifecycle of data starting from the physical storage perspective (e.g. distributed block storage) and concluding on multi-faced data security considerations (e.g. via DLT employment, detection of network intrusions, detection of fraudulent ML model poisoning activities etc.) and metadata management (based on the ontology-based meta-level digital twins' data interfaces). Practically, all elements are supported:</p> <ul style="list-style-type: none"> • Workloads/Services • Workloads/Microservices • Middleware/App server

	<ul style="list-style-type: none"> • Middleware/Connectivity • Middleware/Blockchains • Middleware/M2M • Data/Storages • Data/Data Security • Data/Data streams • Data/Metadata
Things	<p>IoT-NGIN relies on digital twining, employing various data sources as input. Further, it supports IAM at OAuth2.0 level and supports various type of access communication technologies, with an explicit focus on 5G. Elements supported:</p> <ul style="list-style-type: none"> • Smart devices • Digital twin • Communication • IAM • Data sources
Fog-Edge	<p>The IoT-NGIN architecture performs research towards an edge-friendly, ML-powered 5G network configuration management stack, assisted by the MLaaS framework. Further, it caters for the existence of a secure edge cloud execution framework tailored for containers and unikernels. Elements supported:</p> <ul style="list-style-type: none"> • Edge processing • ML/Deep learning
Automation	<p>IoT-NGIN relies on Kubernetes as a resource orchestration framework to manage the secure edge cloud execution framework. Elements supported:</p> <ul style="list-style-type: none"> • Container orchestration
Analytics	<p>IoT-NGIN features (in its application development context), dashboard support for monitoring the platform components and the hosted applications performance. Elements supported:</p> <ul style="list-style-type: none"> • Dashboards

Concluding, the IoT-NGIN architecture fully adheres with the requirements of the meta-architecture and could, therefore, be considered as compliant.

6 Common IoT data models

Cross-organizational and cross-platform data sharing requires common data models which define the syntax and semantics for data exchange. There are several existing data model implementation options that are compatible with the IoT-NGIN requirements and relevant to federated data sovereignty and federated communications.

In IoT-NGIN domain, probably the most important framework to consider is the International Data Spaces Association (IDSA) and its standards as the IDSA is creating a secure and sovereign data economy for Europe and global markets. In context information management frameworks, FIWARE NGSIv2 and ETSI NGSI-LD are potential options for IoT-NGIN. WoT can also be applied by using its building blocks and standardised approach to metadata.

6.1 International Data Spaces

The International Data Spaces (IDS) is a virtual data space that utilises mature standards and technologies in the data economy. According to IDS, a data space is a set of IDSA governed relationships between trusted partners that can be used for secure and sovereign data exchange, certification and governance across diverse organizations. IDS enables cross-organization business processes and transactions and secures the data sovereignty of the owners of the data.

The strategic requirements of IDS are trust, security, data sovereignty, data ecosystems, data apps and standardised interoperability. Research and development based on guidelines such as open development processes, re-use existing components and standardization work. The objectives and activities of IoT-NGIN align with IDS, thus many of the guidelines and solutions of IDS can be adopted by the project.

IDS has developed a reference architecture model, which consists of business, functional, process, information and system layers. In addition, the reference architecture model has three main perspectives of security, certification and governance. The process and information layers combined with certification and governance perspective complement the technologies in IoT-NGIN. For example, the information layer consists of an Information Model with several representations, illustrated in Figure 14.

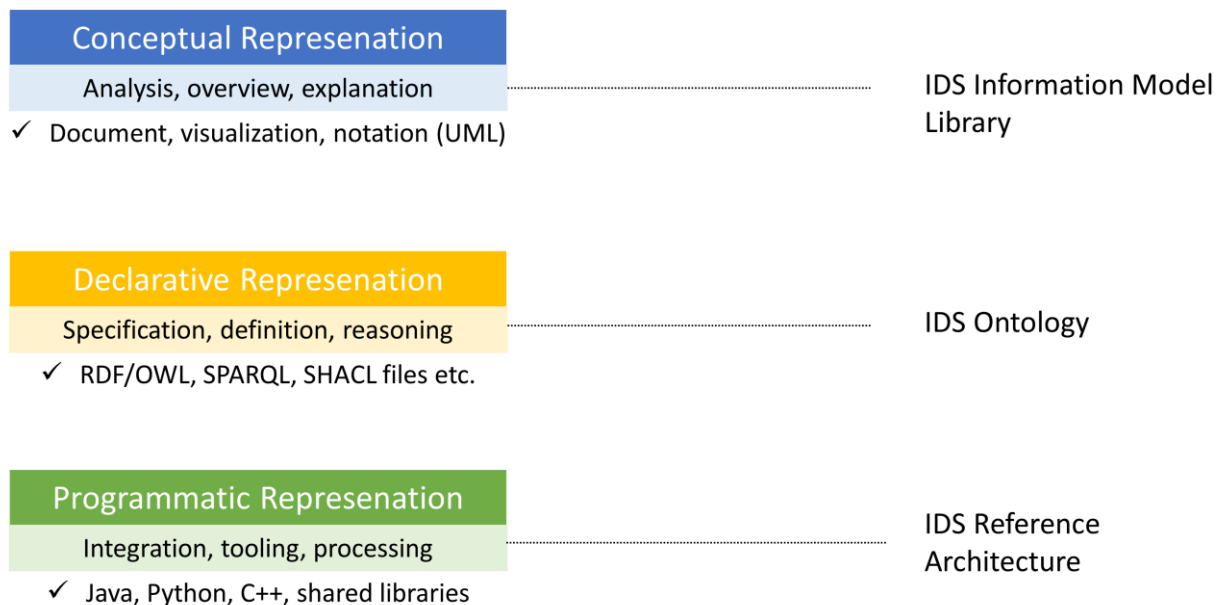


Figure 14: IDS Information Model. Adapted from IDS

The Conceptual Representation is a high-level, conceptual, and technology-agnostic overview analogous to the meta-architecture artefact. The Declarative and Programmatic Representation provide more details. IDS Ontology offers analysis and requirements which are based on W3C Semantic Web standards and other modelling vocabularies such as DCAT or ODRL. Programmatic Representation focuses to provide linkage between commonly used software tools, practices and technologies, and the IDS Ontology.

6.2 W3C Web of Things (WoT)

WoT focuses to harmonise the IoT landscape by developing extending existing and already standardised technologies. WoT is a protocol independent-technology and consists of various building blocks, illustrated in the Figure 15.

WoT Things Description (TD) is an information model and representation format for semantic metadata. The objective for TD is to become "the HTML for smart things". WoT is a protocol independent technology. Binding templates provide a build-in mechanism to define how prevalent protocols such as HTTP, MQTT, HTTP, CoAP can be mapped to the WoT abstractions and interactions in the level of the Thing properties, actions and events. WoT Security and Privacy Guidelines are for general security guidance at the level of the public networks. The guidelines vary from implementation to secure configuration of Things. WoT Scripting API is JavaScript runtime environment for IoT applications, like web browsers. In the context of this study and the developed meta-architecture, WoT is a promising technology that can connect abstract architectural concepts to concrete things and is as a technology-agnostic option [5].

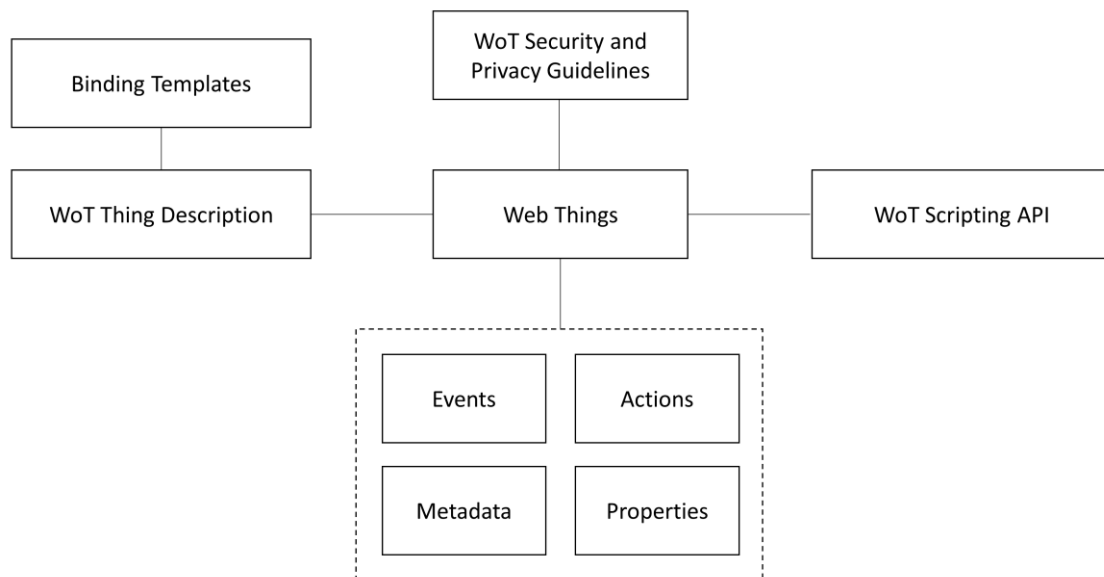


Figure 15: WoT Building Blocks

6.3 FIWARE and ETSI Context Information Management: NGSI-v2 and NGSI-LD

6.3.1 FIWARE NGSI

FIWARE NGSI API integrates platform components within any platform marked as “Powered by FIWARE”. It permits the complying applications to edit or ingest context information. Using GEs is optional, but the Context Broker is required in all FIWARE applications. The Context Figure FIG visualises the GE landscape at high-level. Regarding interfaces, FIWARE NGSI API aligns with the ETSI NGSI-LD standard.

Managing context information is the main objective of FIWARE [4]. In this case, context refers to the operating environment of IoT system and information about its entities, which produce applicable information for system development. The context information consists of varying data sources such as a sensors networks, actuators, services and third-party applications. The following list is an example set of GEs in a Smart Agriculture project [79]:

- Orion Context Broker (OCB): Orion is a C++ implementation of the NGSIv2 REST API. The context information management includes functions for context elements such as build, update, queries, device registration and more. OCB have a lot of useful features, for example cross origin resource sharing (CORS), multi tenancy, and transparent metadata attachment.

- Cygnus is a GE that persists data sources from third-parties and creates historical views of the data. It is based on Apache Flume, a technology that is responsible for design and execution of data collection. A persistence agent operates the data flow through listener receiving data, channel transmitting the data after Flume event transformation, and a sink which stores the Flume events. Examples of supported NGSI-like context data: HDFS, MySQL, MongoDB, Kafka, DynamoDB, PostgreSQL, Elasticsearch, Arcgis.
- Intelligence Data Advanced Solution (IDAS) GE offers an interpreter between the IoT communication protocols and NGSI standard. The available interfaces include LWM2M over CoaP, JSON, UltraLight over HTTP/MQTT, and object linking for OPC-UA.
- Wilma is a PEP Proxy that manages access control in application back ends. Only permitted users can access the GEs. It is often combined with other security components such as Keyrock which is OAuth2.0-based users and devices identity manager, with Single Sign-On and Identity Federation functions.

6.3.2 ETSI NGSI-LD

European Telecommunications Standards Institute (ETSI) defines the NGSI-LD information model which provides context information format and interfacing for NGSI-LD-based applications. The information model defined data representations, APIs and common vocabularies for the API. The context information can map a term string into concept identifiers of URIs. NGSI-LD is capable of several representations, such as entity, property, relationship, context source, and subscription representations. The LD notation of the NGSI protocol stands for linked data since NGSI-LD is built on top of JSON-LD, which is a linked data format of JSON. Linked data applications frequently utilise a concept called "triples". A triple consists of a subject, property, and value. Data graphs can be built out of the triple sets and processed with various graph-based methods or graph processing APIs. JSON-LD offers a formalised representation of linked data through JSON. One effective way to process graphs in general, that applies to JSON, is tree structures. Many dominant programming languages offer built-in tree traversal algorithms that can be extremely efficient. An essential capability of JSON-LD is appending data with context information and transforming the values in a format that allow simple and efficient processing. An optimal outcome is a JSON-like data graph structure, in which conversion between linked data and editable JSON data is trivial [80].

The NGSI-LD Information Model links to high-level Core Meta-model and Cross-Domain Ontology. The lower-level domain-specific ontologies are a point of interest for meta-architecture. For example, IoT-NGIN can potentially use the Smart Appliances REference Ontology (SAREF) Ontology and map it to the NGSI-LD Information Model. Conveniently, SAREF incorporates extensions for the domains of Smart City, Smart Agriculture, Smart Energy and Smart Industry 4.0.

NGSI-LD API can be adapted into versatile architectures and in general, the NGSI-LD includes minimal architectural assumptions. A suitable example regarding the meta-architecture is federated architecture with NGSI-LD. For example, applications that aim to federate existing technologies and domains can exploit the method. In a smart city scenario, distinct departments in an organization of a major city operate unique Context Broker-based NGSI-LD infrastructures. However, smart city applications need to retrieve information from a single source of truth. Entire application domains can be defined as access points through Context

D1.2 - IoT meta-architecture, components, and benchmarking

Brokers, instead of requiring a separate Context Source for every application entity. Information domains are appended to Context Registry and provide the context that can be matched with queries regarding specific topics. As an example, rather than registering individual buildings to the registry, the added information could represent specific building types in a city district. The Federation Broker identifies the domain Context Brokers offering accurate information and forwards the queries and requests. Figure 16 demonstrates the NGSI-LD building blocks of a federated architecture.

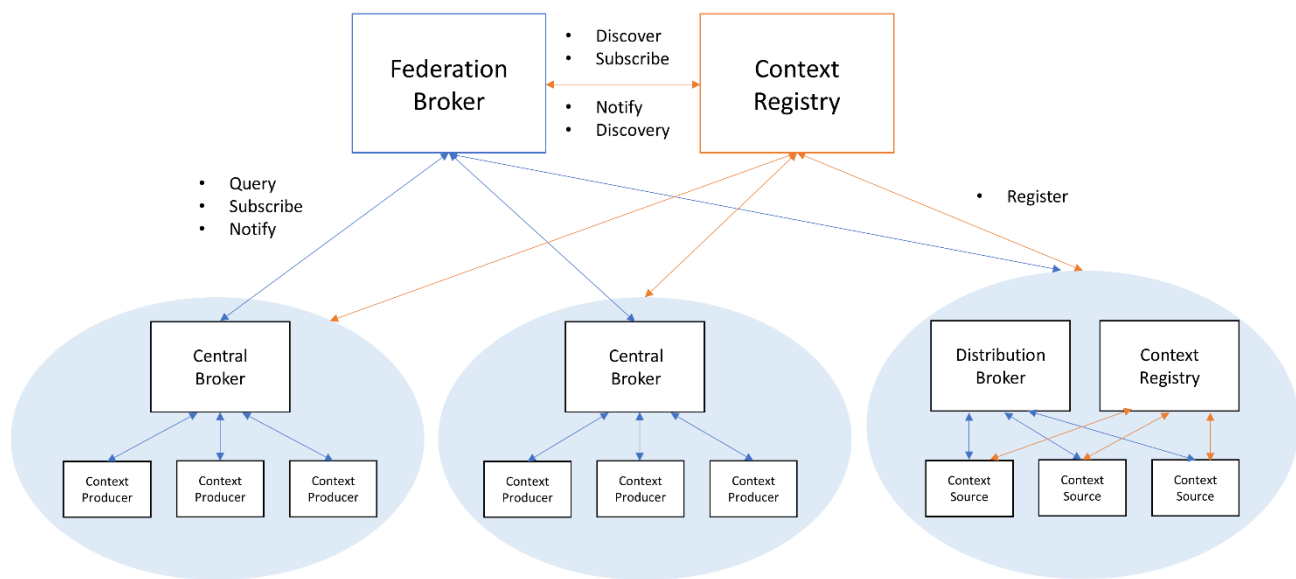


Figure 16: NGSI-LD and federated architecture. Adapted from NGSI-LD.

7 Benchmarking and test reports

In this section, the format of the test reports, as well as the benchmarking methodology to be followed in IoT-NGIN project for the validation of the KPIs will be defined. The benchmarking of the verification framework is heavily based on Quality of Service (QoS) and Quality of Experience (QoE); two metrics that can be utilised to validate the project outcomes in terms of user experience and performance. This section provides the initial approach for the verification framework benchmarking of IoT-NGIN, however, a more elaborated version of the methodology, which will also be fine-tuned to the UCs' requirements, will be provided in D1.3, at a later stage.

The verification framework benchmarking considers technical parameters within the scope of QoS, which assess the performance of the system through the KPIs, and quantifies the user experience via QoE. Other methodologies, such as resource scaling, cross-domain interaction and IoT-NGIN platform testing as a whole, will also be investigated to satisfy Task 1.3 objectives and in light of future developments.

7.1 Methodology

The methodology for the benchmarking of the verification framework is based on the scope and objectives of the project, with particular emphasis been given on each specific UC. The development process considers the Agile model, which consists of multiple cycles of designing, building, testing, and reviewing a product or service continuously. Agile development has numerous advantages, including the nature of the process which is made to constantly adapt and improve the product and the continuous user engagement and feedback [81]. The development of the system is finalised once successful scoring is achieved on User Acceptance Tests (UATs).

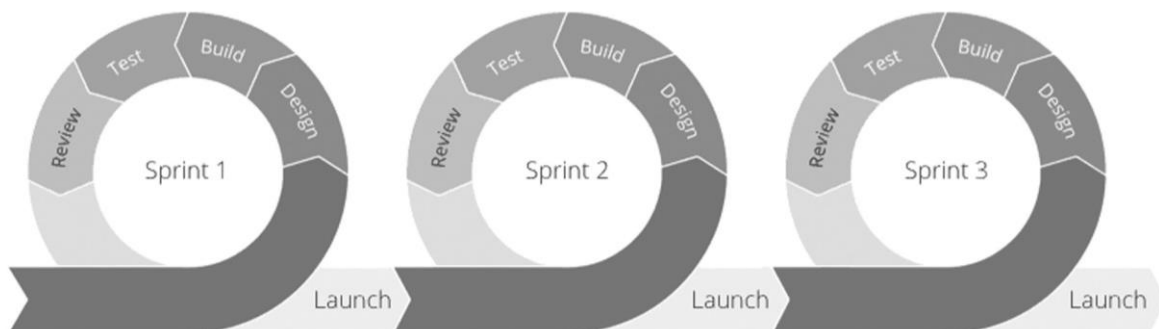


Figure 17: Agile methodology for product evaluation.

7.1.1 Quality of Service

QoS is the tool for measuring the overall performance of a service, with particular focus on the attributes that can be observed by the end-user of the network. Thus, to successfully quantify QoS, parameters such as packet loss, bit rate, transmission delay, and jitter are monitored. Moreover, QoS is utilised for controlling and handling network resources by setting

priorities within the network based on the type of data. In essence, QoS can be defined as a set of actions that allow for the management of network resources and the tool for managing the network with respect to the application, the user, and data flows. Finally, it can be used as a mechanism that ensures a certain level of performance for a data flow. QoS can provide the network administrator with the ability to manage the system effectively, by performing actions such as adjusting the order of handling packets or managing the traffic flow [82].

In this project, the QoS can be evaluated through a number of metrics by assessing the quality of the provided service in system and device levels. Specifically, metrics that can be utilised for the assessment of QoS on a system level are the following:

1. System quality: the quality of the sensor network can be evaluated as a whole based on the quality of the data provided after a query.
2. Delay: measured during data collection from nodes.
3. Bandwidth: Assessing the capacity of a sensor network in sending data over a link.
4. System lifetime: Indicates the longevity of the system.
5. Resource optimisation: A metric that is defined by the ability of the system to efficiently allocate its limited resources within a society and thus optimise its resource utilisation by maximising the social welfare.

On a device level, the following metrics can be used:

1. Quality of IoT devices: Assessment of the accuracy and sensitivity of the measurements provided by the IoT devices.
2. Energy consumption: Assessment of the energy consumption of the system, which is particularly important for wireless sensor networks (WSN), as the lifetime of the devices is dependent on it.
3. Bandwidth: Measurement of the bandwidth usage of the IoT devices.
4. Data volume: Amount of data generated by IoT devices.
5. Trustworthiness: Assesses the reliability of the sensor in the scope of delivering accurately and timely measurements. This metric is associated with the quality metric mentioned earlier.

An example template for QoS evaluation is presented in Table 37. For IoT-NGIN, the QoS parameters will be defined once all the components for each UC have been identified and characterised.

Table 37: Sample QoS metrics.

Traffic class	Technology attributes	Time			Preciseness			Accuracy
		Response time expected by users	Delay	Jitter	Data rate	Required bandwidth	Loss rate	Error rate

7.1.2 Quality of Experience

QoE is a metric based on the user satisfaction or annoyance, which focuses on the overall service experience holistically. According to the European Telecommunications Standard Institute (ETSI), QoE is defined as a method for measuring performance according to users based on subjective and objective psychological measurement, for the use of a product or a service [83]. The concept of QoE has emerged from the field of telecommunications, in an effort to quantify the user experience while using a service and therefore, to understand the user's overall quality requirements. Although QoS remains a powerful tool for evaluating a service in terms of performance, current services and functionalities that are offered within the IoT domain are user centric. Thus, QoE becomes essential for obtaining feedback by the end-user [84].

The following factors can be evaluated through QoE:

1. Usability: The rate at which users can use the product and achieve their goals effectively and efficiently.
2. Usefulness: The ability of the product to fulfil the user's needs and/or preferences.
3. Transparency: The ability of users to view the performed actions.
4. Effectiveness: The ability of users to complete their tasks.
5. Efficiency: The effort that is required by the users to complete their tasks.
6. Accessibility: The ability of multiple groups of people to use the product.
7. Personalisation: Individualisation of the product based on user's needs and predefined preferences.
8. Learnability: The effort required by the end-user to use the product.

7.2 Implementation of methodology for benchmarking

Table 38 lists all the UCs that will be examined under IoT-NGIN project, as they have been defined in D1.1. The UCs are spread across four verticals, namely smart cities, smart agriculture, Industry 4.0 and smart energy. In spite of the common goal the project has on the aforementioned areas, each vertical has its own particularities and hence, the benchmarking has to be designed to fulfil the needs of each one and in some cases, to be tailored for a specific UC. Moreover, it also depends on the specific group of people involved in each UC, as the actors and users for each case could be regulatory authorities or companies, managerial personnel and employees, general public, or a combination of these categories.

Table 38: UCs of IoT-NGIN.

Use case	Title	Vertical
UC1	Traffic Flow Prediction & Parking prediction	Smart cities
UC2	Crowd management	
UC3	Co-commuting solutions based on social networks	

UC4	Crop diseases prediction. Smart irrigation and precision aerial spraying	Smart agriculture
UC5	Sensor aided crop harvesting	
UC6	Human-centred safety in a self-aware indoor factory environment	
UC7	Human-centred augmented reality assisted build-to-order assembly	Industry 4.0
UC8	Digital powertrain and condition monitoring	
UC9	Move from reacting to acting in smart grid monitoring and control	Smart energy
UC10	Driver-friendly dispatchable EV charging	

UC1 is considered as an example in this section, which aims at efficient traffic flow prediction and parking availability, in order to decrease traffic jams and bottlenecks. To this end, in this UC a model will be developed that:

- Aids the driver in choosing a less-trafficked road.
- Assists in finding a parking spot by providing information on available parking locations.
- Demonstrate the application of deep learning technologies for advanced traffic flow prediction including unpredictable conditions like weather, delays and accidents.

For this UC three KPIs shall be satisfied, which have already been defined in D1.1 and are presented in Table 39. The QoS with regards to UC1 will be based on the specific KPIs, where parameters such as the performance of the monitoring devices with respect to system quality, lifetime, bandwidth etc. are evaluated and compared against the requirements and needs of the KPIs.

Table 39: KPIs for UC1.

KPI ID	Name	Description	Method of measurement	Target
KPI_UC1_1	Real-time monitoring	Improve efficiency and traffic congestions in twin smart cities	Logs and analysis	$\geq 20\%$
KPI_UC1_2	Cross-border data models	Number of proposed cross-border data models	Number of data models	> 4
KPI_UC1_3	Data sources analysis	Number of different types of sensors' data to be analysed	Number of different sensors used	> 6

On the other hand, as mentioned earlier QoE does not consider the performance of the product from a technical point-of-view, but rather focuses on the experience and

satisfaction of the end-user. Hence, questions in the form of a questionnaire or an online Q&A will be implemented, in order to capture the experience of the end-user. The format of the test report for QoE in the form of a questionnaire is shown in Table 40, where example questions for UC1 are presented.

Table 40: Example of test report format for QoE.

1. Please rate the reduction experienced in traffic.									
1	2	3	4	5	6	7	8	9	10
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. Please rate the time reduction you experienced in finding a parking spot.									
1	2	3	4	5	6	7	8	9	10
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. Please rate the less-trafficked proposed routes.									
1	2	3	4	5	6	7	8	9	10
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. Please rate the reduction of traffic at unpredictable conditions (e.g. weather, delays, accidents etc.)									
1	2	3	4	5	6	7	8	9	10
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

The rating of the experience could be based on a rating system with scores from poor to excellent. Finally, any material related to the QoE will be compliant to the Data Management policies of the IoT-NGIN project. In summary, the methodology presented in this section for the benchmarking of the verification framework, once applied, will be reassessed and thus, parameters will be reevaluated in order to be updated with developments. Therefore, in case where new requirements need to be established within the framework or existing requirements need to be altered, the benchmarking will be readjusted to include all changing factors.

8 Conclusions

This document constitutes Deliverable “D1.2: IoT meta-architecture, components and benchmarking” of the European H2020-ICT-2018-20 “IoT-NGIN: Next Generation IoT as part of Next Generation Internet” project. The document defines the initial version of IoT-NGIN meta-architecture, supporting a multitude of use cases across domains, and validated in the IoT-NGIN Living Labs.

The aim of meta-architecture work in IoT-NGIN is to provide a research-informed framework for designing and implementing IoT solutions in different usage scenarios. The IoT-NGIN meta-architecture is based on thorough analysis of current state-of-the-art IoT platforms and standards, combined with requirements and expert knowledge provided by partner organizations. The analysis of current state-of-the-art approaches covers key open-source and commercial IoT platforms and has been further extended by EU research perspective covering most relevant EU research projects in recent years.

The meta-architecture presented in this document collects key quality requirements, architectural patterns and high-level system components aiming to provide an overall framework for individual system implementations. The motivation is to enable reuse of existing IoT technologies and solutions to new domains and assist structured, informed IoT platform design. The IoT-NGIN meta-architecture is designed around four key artifacts, namely the IoT Architectural Pattern Vertical, the Domain Horizontal, the Quality Vertical, and the Element View. The Element View provides a general technological view to the meta-architecture, including technological components organized in functional groups, thus indicating the role and interrelations among IoT functionalities.

Moreover, the IoT-NGIN architecture is presented as a reference instantiation of the IoT-NGIN meta-architecture. This architecture includes fine-grained definition of the functional components within the respective functional groups, and defines their distribution in IoT and Edge/Cloud nodes. The IoT-NGIN architecture will be instantiated in five IoT-NGIN LLs and is expected to be further extended with the contributions of new partners selected via the IoT-NGIN Open Calls.

It should be noted, that the IoT-NGIN meta-architecture is not static. As IoT and related technologies mature, some elements may get outdated, and as more implementation experience and further requirements are gathered from use cases and living labs over time, the meta-architecture also needs to evolve over time. Consequently, the meta-architecture has been designed to be open and extensible to make room for new technologies and computing paradigms. The continuous evolution and improvement of the meta-architecture will be part of IoT-NGIN project work in the future.

9 References

- [1] Fortune Business Insights , "Internet of Things (IoT) Market Worth USD 1463.19 Billion by 2027 Backed by Rising Awareness Regarding Precision Farming to Aid Market Growth, says Fortune Business Insights," 2021. [Online]. Available: <https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307>.
- [2] International Data Corporation, "European IoT Spending to Exceed \$200 Billion in 2021 as Companies Start Moving to the Next Stage of Recovery, According to IDC," 2021. [Online]. Available: <https://www.idc.com/getdoc.jsp?containerId=prEUR147929621>.
- [3] P. Pierleoni, R. Concetti, A. Belli and L. Palma, "Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison," *IEEE Access*, vol. 8, pp. 5455-5470, 2019.
- [4] FIWARE, "Components," 2021. [Online]. Available: <https://www.fiware.org/developers/catalogue/>.
- [5] W3C, "Web of Things (WoT) Thing Description," 2020. [Online]. Available: <https://www.w3.org/TR/wot-thing-description/>.
- [6] DeviceHive, "Documentation," 2021. [Online]. Available: <https://docs.devicehive.com/docs>.
- [7] ThingsBoard, "ThingsBoard Cloud Documentation," 2021. [Online]. Available: <https://thingsboard.io/docs/paas/>.
- [8] T. KaaloT, "Kaa IoT Platform Features for Enterprise IoT Projects," 2021. [Online]. Available: <https://www.kaaiot.com/overview>.
- [9] OpenRemote, "Product," 2021. [Online]. Available: <https://openremote.io/product/>.
- [10] M. Labs, "Overview," 2021. [Online]. Available: <https://mainflux.readthedocs.io/en/latest/>.
- [11] A. Microsoft, "Azure IoT," 2021. [Online]. Available: <https://azure.microsoft.com/en-us/overview/iot/>.
- [12] S. Amazon Web, "AWS IoT," 2021. [Online]. Available: https://aws.amazon.com/iot/?nc2=h_ql_prod_it.
- [13] Google, "Google Cloud IoT Core documentation," 2021. [Online]. Available: <https://cloud.google.com/iot/docs>.
- [14] P. ThingWorx, "ThingWorx Connect Product Brief," 2021. [Online]. Available: <https://www.ptc.com/en/resources/iiot/product-brief/thingworx-connect>.

- [15] IBM, "Internet of Things on IBM Cloud," 2021. [Online]. Available: <https://www.ibm.com/cloud/internet-of-things>.
- [16] Software AG, "Interfacing Devices," 2021. [Online]. Available: <http://www.cumulocity.com/guides/concepts/interfacing-devices/>.
- [17] Particle, "Device OS," 2021. [Online]. Available: <https://docs.particle.io/tutorials/device-os/device-os/>.
- [18] European Commission, "IoT European Large-Scale Pilots Programme," 2018. [Online]. Available: <https://european-iot-pilots.eu/projects/>. [Accessed 2021].
- [19] CREATE-IoT Project, "CRoss fErtilisation through AlignmenT, synchronisation and Exchanges for IoT," 2021. [Online]. Available: <https://european-iot-pilots.eu/project/create-iot/>.
- [20] SynchroniCity, "SynchroniCity | A universal approach to developing, procuring and deploying IoT- and AI-enabled services," 2021. [Online]. Available: <https://synchronicity-iot.eu/>.
- [21] MONICA Project, "MONICA - MONICA - Sound and security applications for large, open-air events," 2021. [Online]. Available: <https://www.monica-project.eu/>.
- [22] AUTOPILOT, "Homepage - Autopilot," 2021. [Online]. Available: <https://autopilot-project.eu/>.
- [23] ACTIVAGE Project, "ACTIVAGE Project : Internet of Things (IoT) for ageing well," 2021. [Online]. Available: <http://www.activageproject.eu/>.
- [24] IoF2020, "Internet of Food and Farm 2020 - IoF2020," 2021. [Online]. Available: <https://www.iof2020.eu/>.
- [25] GAIA-X, "Gaia-X: A Federated Secure Data Infrastructure," 2021. [Online]. Available: <https://www.gaia-x.eu/>.
- [26] GAIA-X, "GAIA-X: Technical Architecture," Jun 2020. [Online]. Available: https://www.data-infrastructure.eu/GAIAX/Redaktion/EN/Publications/gaia-x-technical-architecture.pdf?__blob=publicationFile&v=5.
- [27] AGILE Project, "AGILE – Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts," 2021. [Online]. Available: <https://www.agile-project.eu/>.
- [28] AGILE, "D3.1 requirements specification sw architecture, V1.0," Jun 2016. [Online]. Available: <https://cloud.google.com/iot/docs>.
- [29] U4IoT, "User Engagement for Large Scale Pilots in the Internet of Things," 2021. [Online]. Available: <https://u4iot.eu/>.

- [30] SOFIE Project, "Secure Open Federation for Internet Everywhere," 2021. [Online]. Available: <https://www.sofie-iot.eu/>.
- [31] N.-5. Project, "Home - 5GPPP," 2021. [Online]. Available: <http://www.nrg5.eu/>.
- [32] OPEN DEI Project, "Aligning Reference Architectures, Open Platforms and Large-Scale Pilots in Digitising European Industry," 2021. [Online]. Available: <https://www.opendei.eu/>.
- [33] SOGNO Project, "Service Oriented Grid for the Network of the Future | SOGNO energy," 2021. [Online]. Available: <https://www.sogno-energy.eu/>.
- [34] INTER-IoT Project, "INTER-IoT - Interoperability Internet of Things," 2021. [Online]. Available: <https://inter-iot.eu/>.
- [35] INTER-IoT, "Final Reference IoT Platform Meta-architecture and Meta-data Model, D4.2," 2018.
- [36] AI4EU Project, "Home | AI4EU," 2021. [Online]. Available: <https://www.ai4europe.eu/>.
- [37] R. Mark, *Software Architecture Patterns*, O'Reilly Media, Inc., 2015.
- [38] S. G. Tzafestas, "The Internet of Things: A Conceptual Guided Tour," *European Journal of Advances in Engineering and Technology*, vol. 5, no. 10, pp. 745-767, 2018.
- [39] R. Stackowiak, "Modern IoT Architecture Patterns," in *Azure Internet of Things Revealed: Architecture and Fundamentals*, Apress, 2019.
- [40] S. Bera, S. Misra and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994-2008, 2017.
- [41] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. U. Rasool and W. Dou, "Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1761-1804, 2020.
- [42] M. Behzad, M. Abdullah, M. T. Hassan, Y. Ge and M. A. Khan, "Performance Optimization in IoT-Based Next-Generation Wireless Sensor Networks," in *Transactions on Computational Collective Intelligence XXXIII. Lecture Notes in Computer Science*, 2019.
- [43] 3GPP, "Technical Specification Group Services and System Aspects; Management and Orchestration; 5G Network Resource Model (NRM); Stage 2 and Stage 3 (Release 16)," Jun. 2019. [Online]. Available: https://www.3gpp.org/ftp/TSG_SA/WG5_TM/TSGS5_125/SA_84/28541-g10.doc. [Accessed Sep. 2021].
- [44] Microsoft, "Microsoft Azure IoT Reference Architecture," 2018.
- [45] R. Nisse, J. L. Hernández-Ramos, S. N. Matheu-García, G. Baldini, A. Skarmeta, V. Siris, D. Lagutin and P. Nikander, "An Interledger Blockchain Platform for Cross-Border

Management of Cybersecurity Information," *IEEE Internet Computing*, vol. 24, no. 3, pp. 19-29, 2020.

- [46] A. Pérez, G. Moltó, M. Caballer and A. Calatrava, "Serverless computing for container-based architectures," *Future Generation Computer Systems*, vol. 83, pp. 20-59, 2018.
- [47] A. Preukschat and D. Reed, *Self-Sovereign Identity: Decentralized digital identity and verifiable credentials*, Manning, 2021.
- [48] Industrial Internet Consortium, *Digital Twins for Industrial Applications*, 2020.
- [49] J. Autiosalo, J. Vepsalainen, R. Viitala and K. Tammi, "A Feature-Based Framework for Structuring Industrial Digital Twins," *IEEE Access*, vol. 8, 2020.
- [50] E. Cavalcante, M. P. Alves, T. Batista, F. C. Delicato and P. F. Pires, "An analysis of reference architectures for the internet of things," in *2015 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures (CobRA)*, Montreal, QC, Canada, 2015.
- [51] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. van Kranenburg, S. Lange and S. Meissner, *Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model*, Springer, 2013.
- [52] A. Poniewierski, "How the IoT and data monetization are changing business models," Ernst & Young Global Limited, 20 Feb. 2019. [Online]. Available: https://www.ey.com/en_gl/consulting/how-the-iot-and-data-monetization-are-changing-business-models. [Accessed 08 Sep. 2021].
- [53] Cloud Native Computing Foundation (CNCF), "CNCF Cloud Native Definition v1.0," 11 Jun. 2018. [Online]. Available: <https://github.com/cncf/toc/blob/master/DEFINITION.md>. [Accessed 08 Sep. 2021].
- [54] "Unikernels: Rise of the Virtual Library Operating System," . [Online]. Available: <http://queue.acm.org/detail.cfm?id=2566628>. [Accessed 8 9 2021].
- [55] "Production-Grade Container Orchestration," The Linux Foundation, Sep. 2021. [Online]. Available: <https://kubernetes.io/>. [Accessed 08 Sep. 2021].
- [56] Cloud Native Computing Foundation, "Cloud Native Interactive Landscape," 08 (dynamic) Sep. 2021. [Online]. Available: <https://landscape.cncf.io/?project=graduated>. [Accessed 08 Sep. 2021].
- [57] "Containerd: An industry-standard container runtime with an emphasis on simplicity, robustness and portability," Sep. 2021. [Online]. Available: <https://containerd.io/>. [Accessed 08 Sep. 2021].
- [58] "Policy-based control for cloud native environments," Sep. 2021. [Online]. Available: <https://openpolicyagent.org>. [Accessed 08 Sep. 2021].

- [59] "Linkerd: A different kind of service mesh," Sep. 2021. [Online]. Available: <https://linkerd.io/>. [Accessed 08 Sep. 2021].
- [60] "The Update Framework," Sep. 2021. [Online]. Available: <https://theupdateframework.io/>. [Accessed 08 Sep. 2021].
- [61] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125-1142, 2017.
- [62] M. Noura, M. Atiquzzaman and M. Gaedke, "Interoperability in Internet of Things Infrastructure: Classification, Challenges, and Future Work," in *International Conference on Internet of Things as a Service*, 2017.
- [63] A. Javed, A. Malhi, T. Kinnunen and K. Främling, "Scalable IoT Platform for Heterogeneous Devices in Smart Environments," *IEEE Access*, vol. 8, pp. 211973-211985, 2020.
- [64] A. Shukla and Y. Simmhan, "Benchmarking Distributed Stream Processing Platforms for IoT Applications," in *Technology Conference on Performance Evaluation and Benchmarking*, 2017.
- [65] H2020 IoT-NGIN Consortium, "Deliverable D3.1: Enhancing Deep learning/reinforcement learning," 2021.
- [66] Microsoft, "Ambassador pattern," 23 Jun. 2017. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/patterns/ambassador>. [Accessed Sep. 2021].
- [67] Microsoft, "Choreography pattern," 24 Feb. 2020. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/patterns/choreography>. [Accessed Sep. 2021].
- [68] ETSI, "Open Source MANO (OSM)," [Online]. Available: <https://www.etsi.org/technologies/open-source-mano>. [Accessed Sep. 2021].
- [69] "Rust is a systems programming language," , [Online]. Available: <https://www.rust-lang.org>. [Accessed 21 9 2021].
- [70] The Linux Foundation, "Rook: Open-Source, Cloud-Native Storage for Kubernetes," [Online]. Available: <https://rook.io/>. [Accessed Sep. 2021].
- [71] Ceph Foundation, "Ceph: The Future of Storage," [Online]. Available: <https://ceph.io/en/>. [Accessed Sep. 2021].
- [72] E. . Bisong, "Deploying an End-to-End Machine Learning Solution on Kubeflow Pipelines," , 2019. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-4470-8_47. [Accessed 21 9 2021].

- [73] M. . Salvaris, D. . Dean and W. H. Tok, "Generative Adversarial Networks," *arXiv: Machine Learning*, vol. , no. , pp. 187-208, 2018.
- [74] S. Bourou, A. El Saer, T.-H. Velivassaki, A. Voulkidis and T. Zahariadis, "A Review of Tabular Data Synthesis Using GANs on an IDS Dataset," *Information*, p. 14, 14 Sep. 2021.
- [75] "Apache Kafka at GitHub," , . [Online]. Available: <https://github.com/apache/kafka>. [Accessed 21 9 2021].
- [76] The Linux Foundation, "Kubernetes API," [Online]. Available: <https://kubernetes.io/docs/reference/kubernetes-api/>. [Accessed Sep. 2021].
- [77] ETSI OSM, "Welcome to Open Source MANO's documentation," 2020. [Online]. Available: <https://osm.etsi.org/docs/user-guide/>. [Accessed Sep. 2021].
- [78] The Linux Foundation, "Prometheus: Monitoring system & time series database," [Online]. Available: <https://prometheus.io/>. [Accessed Sep. 2021].
- [79] M. Muñoz, J. D. Gil, L. Roca, F. Rodríguez and M. Berenguel, "An IoT Architecture for Water Resource Management in Agroindustrial Environments: A Case Study in Almería (Spain)," *Sensors*, vol. 3, 2020.
- [80] E. G. C. 009, "Context Information Management (CIM)," ETSI Industry Specification Group, 2019.
- [81] Y. Y. Jusoh, S. Abdullah, I. M. Ali, M. H. M. Noh, M. H. Mazlan, C. S. Bouh and T. Z. Sheng, "Adoption of Agile Software Methodology Among the SMEs Developing IoT Applications," in *6th International Confere4nce on Research and Innovation in Information Systems (ICRIIS)*, 2019.
- [82] M. S. a. G. Baranwal, "Quality of Service (QoS) in Internet of Things," in *3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, 2018.
- [83] E. T. 1. 6. V1.0.2, "Human Factors (HF); Quality of Experience (QoE) requirements for real-time communication services," Sophia Antipolis, 2010.
- [84] O. A. a. J. M. L.G.M. Ballesteros, "Quality of Experience (QoE) in the smart cities context: An initial analysis," in *2015 IEEE First International Smart Cities Conference (ISC2)*, 2015.