nephele

A Lightweight Software Stack and Synergetic Meta-Orchestration Framework for the Next Generation Compute Continuum

D2.2 - NEPHELE Reference Architecture Final Specification

Document Identification			
Status	Final	**Due Date**	29/02/2024
Version	1.0	**Submission Date**	08/03/2024

Related WP	WP2	**Document Reference**	D2.2
Related Deliverable(s)	D3.1, D3.2, D4.1, D4.2, D5.1	**Dissemination Level (*)**	PU
Lead Participant	NTUA	**Lead Author**	Anastasios Zafeiropoulos
Contributors	NTUA, CNIT, SIEMENS, ATOS, INRIA, UOM, ODINS, SMILE, ININ, ECL, ERCIM, ZHAW	**Reviewers**	Giacomo Genovese (CNIT), Adriana Arteaga Arce (INRIA)

Keywords:
Meta-Orchestration, IoT Software Stack, Viewpoints, NEPHELE Architecture, Development environment, Dashboard

Document Information

List of Contributors	
Name	Partner
Anastasios Zafeiropoulos	NTUA
Dimitrios Spatharakis	NTUA
Ioannis Dimolitsas	NTUA
Nikos Filinis	NTUA
Eleni Fotopoulou	NTUA
Constantinos Vassilakis	NTUA
Manolis Katsaragakis	NTUA
Dimosthenis Masouros	NTUA
Ioannis Tzanettis	NTUA
Symeon Papavassiliou	NTUA
Rafael Marín Pérez	ODINS
Alejandro Arias Jiménez	ODINS
Panagiotis Papadimitriou	UOM
Lefteris Mamatas	UOM
Ilias Sakellariou	UOM
George Papathanail	UOM
Leonardo Militano	ZHAW
Giovanni Toffetti	ZHAW
Guillermo Gomez Chavez	ATOS
Adriana Arteaga Arce	INRIA
Giacomo Genovese	CNIT
Darko Anicic	SIEMENS
Rudolf Sušnik	ININ
Marco Jahn	ECL
Dave Raggett	ERCIM
Jonathan Rivalan	SMILE

Document History			
Version	Date	Change editors	Changes
0.1	01/11/2023	Anastasios Zafeiropoulos	ToC preparation and first draft
0.2	16/11/2023	Anastasios Zafeiropoulos, Manolis Katsaragakis	Editing of Sections 1, 2 and 3
0.3	20/12/2023	All contributors	Editing of Section 4
0.4	31/01/2024	All contributors	Editing of Section 5
0.5	07/02/2024	Anastasios Zafeiropoulos	Full version of the deliverable available for internal review
0.6	15/02/2024	Giacomo Genovese, Adriana Arteaga Arce	Comments by the internal review
0.7	07/03/2024	Anastasios Zafeiropoulos, Manolis Katsaragakis, Dimitrios Spatharakis	Updated version with revisions
1.0	08/03/2024	Symeon Papavassiliou	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Anastasios Zafeiropoulos (NTUA)	07/03/2024
Internal reviewers	Giacomo Genovese (CNIT), Adriana Arteaga (INRIA)	07/03/2024
Project Coordinator	Symeon Papavassiliou (NTUA)	08/03/2024

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation /Acronym	Description
5G	Fifth Generation
AD	Architecture Description
AI	Artificial Intelligence
API	Application Programming Interface
ASP	Application Service Provider
CCNM	Computing Continuum Network Manager
CEP	Cloud-Edge Provider
CM	Cluster Manager
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
cVO	Composite Virtual Object
DL	Deep Learning
DT	Digital Twin
Dx,y	Deliverable number y belonging to WP x
E2E	End-to-End
EC	European Commission
FR	Functional Requirement
HA	High-Availability
HDA	Hyper Distributed Application
HDAG	Hyper Distributed Application Graph
HTTP	HyperText Transfer Protocol
HW	Hardware
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IPvx	Internet Protocol Version x
IoT	Internet of Things
IPR	Intellectual Property Rights
ISO	International Organization for Standardization
LAN	Local Area Network
LwM2M	Lightweight Machine-to-Machine
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NFR	Non-Functional Requirement
NP	Network Provider

Abbreviation /Acronym	Description
OIDC	OpenID Connect
OIDC4CI	OpenID Connect for Credential Issuance
OMA	Open Mobile Alliance
QoS	Quality of Service
SLA	Service Level Agreement
SLO	Service Level Objective
SMO	Synergetic Meta-Orchestrator
SW	Software
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UC	Use Case
UDP	User Datagram Protocol
UI	User Interface
UX	User Experience
VC	Verifiable Credentials
VO	Virtual Object
VOStack	Virtual Object Software Stack
VPN	Virtual Private Network
VXLAN	Virtual Extensible LAN
W3C	World Wide Web Consortium
WP	Work Package

Executive Summary

NEPHELE is a Research and Innovation Action with a duration of 36 months involving 17 partners from 9 countries and several sectors. The project aims to "enable the efficient, reliable and secure end-to-end orchestration of hyper-distributed applications over programmable infrastructure that is spanning across the compute continuum from Cloud-to-Edge-to-IoT, removing existing openness and interoperability barriers in the convergence of IoT technologies against cloud and edge computing orchestration platforms, and introducing automation and decentralized intelligence mechanisms powered by 5th Generation (5G) and distributed Artificial Intelligence (AI) technologies".

This document presents the major architectural description of the NEPHELE's platform. More specifically, the architectural description relies on the utilization of the ISO/IEC/IEEE 42010 standard, aiming to present a conceptual foundation for expressing, communicating and reviewing architectures and specifying requirements that apply to architecture descriptions. Initially, the NEPHELE ecosystem and the major objectives are detailed. Next, the different stakeholders, their role and their major concerns are presented in depth, aiming to derive their different expectations from NEPHELE's system. Following, a detailed analysis of the different views, viewpoints and their correlation with the stakeholders is shown. The set of the views and viewpoints relies on a set of functional and non-functional requirements that have to be fulfilled, considering their criticality in NEPHELE's system operation and stability, along with their design challenges and difficulties. Following, the architectural description of the NEPHELE project is derived and the major design principles for the effective co-design of hardware/software components and the interoperability among them is presented.

This document will be utilized as the basis for the development of the VOStack in WP3, the synergetic orchestration mechanisms and the development environment in WP4, and the integrated NEPHELE platform in WP5.

1. Introduction

NEPHELE is a Research and Innovation Action (RIA) project funded by the Horizon Europe programme under the topic "Future European platforms for the Edge: Meta Operating Systems". The NEPHELE's vision is to enable the efficient, reliable and secure end-to-end orchestration of hyper-distributed applications over a programmable infrastructure that is spanning across the compute continuum from IoT-to-edge-to-cloud. This deliverable reports on the activities of Work Package 2 (WP2). WP2 is devoted to collecting the requirements for intelligent IoT device management and coordination and synergetic orchestration of cloud and edge computing applications to provide the breakthrough reference architecture of NEPHELE. More specifically, this deliverable focuses on the Task 2.4, which aims to specify and detail the NEPHELE reference architecture, functionalities and mechanisms, based on a set of requirements derived in Tasks 2.1-2.3 and analyzed in D2.1 [11]. The deliverable builds upon the work in D2.1 where a set of definitions for the main components of the NEPHELE ecosystem are provided, including a set of requirements that have to be supported per component.

In this deliverable, we initially present an overview of the vision of the NEPHELE ecosystem and its major objectives (Section 2). The architectural description of the NEPHELE platform relies on the ISO/IEC/IEEE 42010, thus an overview of the standard and its integration on NEPHELE is analyzed (Section 3). The ISO/IEC/IEEE 42010 standard requires an in-depth definition of stakeholders, concerns, views and viewpoints, which addresses the creation, analysis and sustainment of architectures of systems through the use of architecture descriptions. Thus, the different stakeholders, their role and their major concerns are presented in depth, aiming to derive their different expectations from the NEPHELE's system alongside with a detailed analysis of the different views, viewpoints and their correlation with the stakeholders (Section 4). Next, the final architecture description of NEPHELE is presented (Section 5). The major objective of this deliverable is to specify the NEPHELE's platform architecture regarding the functionality, scalability, interoperability, robustness and seamless collaboration among the dominant NEPHELE components, i.e. i) the Meta-Orchestration platform, ii) the Virtual Object Stack (VOStack), iii) the Dashboard and Development environment for the deployment of Hyper Distributed Applications (HDAs).

The work achieved in this deliverable will serve as a base reference document to the other WPs and deliverables of the project since it gathers main information useful for the project.

2. NEPHELE Vision and Objectives

The advancement of Internet of Things (IoT) and Edge Computing technologies is progressing swiftly. This rapid evolution is reshaping businesses and daily lives, introducing solutions tailored for diverse industrial sectors and laying the groundwork for a fully interconnected world. Simultaneously, this evolution aligns with the increasing diversity of IoT technologies. This diversity encompasses the creation of various intelligent IoT devices, the support for diverse communication protocols, and the conceptualization of distinct information models for semantically representing entities/resources in the IoT landscape. These trends underscore the necessity for innovative architectural approaches that inherently support the convergence and integration of existing and evolving IoT and edge computing technologies. To effectively manage data processing and analysis in this distributed environment, novel hyper-distributed applications (HDAs) increasingly embrace microservices-based and cloud-native computing technologies. Distributed computing principles are also evolving their lifecycle orchestration paradigms to efficiently utilize resources across the spectrum from IoT-to-Edge-to-Cloud.

Two primary challenges are identified and aimed to be tackled in NEPHELE. Firstly, there is a need for the convergence of IoT technologies based on innovative architectural approaches. These approaches must ensure continuous openness and interoperability across a myriad of existing and emerging solutions, models, and devices. Simultaneously, they should enable analytics to assess the lifecycle costs, measured in time and resources, ranging from seconds to CO2 emissions. Secondly, there is a requirement for the establishment of an integrated meta-orchestration environment for HDAs. This environment should facilitate synergy between cloud and edge computing orchestration platforms, optimizing the end-to-end deployment of applications and data provision throughout the continuum.

Addressing these challenges, the NEPHELE project endeavors to introduce two core innovations:

- An IoT and Edge computing software stack -called as VOStack- designed to leverage the virtualization of IoT devices at the edge infrastructure. This stack supports openness and interoperability in a device-independent manner. Through this software stack, management of diverse IoT devices and platforms can be unified, eliminating the need for middleware platforms. Additionally, edge computing functionalities can be dynamically provided to efficiently support IoT applications.
- A synergistic meta-orchestration framework for managing coordination between cloud and edge computing orchestration platforms. This framework employs high-level scheduling supervision and definition, grounded in a "system of systems" approach.

These two core innovations are accompanied by the NEPHELE Dashboard and Development Environment, offering a set of tools to application developers and providers for the adoption and usage of the developed innovations.

3. Overview of ISO/IEC/IEEE 42010 and Integration on NEPHELE System

ISO/IEC/IEEE 42010, commonly referred to as the "Systems and software engineering - Architecture Description" standard is a comprehensive framework that plays a pivotal role in the discipline of system and software architecture. This international standard, jointly developed by ISO (International Organization for Standardization), IEC (International Electrotechnical Commission), and IEEE (Institute of Electrical and Electronics Engineers), provides a unified and well-defined approach for describing architectures of complex systems. It's a foundational document that establishes standardized terminology, concepts, and principles, promoting clear communication and understanding among stakeholders involved in architecting, designing, and managing systems. ISO/IEC/IEEE 42010 lays the groundwork for effective architecture description, offering a structured way to express and document the essential elements, relationships, and properties of a system's architecture.

At its core, ISO/IEC/IEEE 42010 defines architecture as the fundamental characteristics of a system, encompassing its elements, their interconnections, and the guiding principles governing its design and evolution. The standard recognizes the diverse interests of stakeholders in a system and introduces the concept of viewpoints, each tailored to address the specific concerns of particular stakeholders. These viewpoints guide the creation of views, which are representations of the system from distinct perspectives. By organizing architectural information into coherent viewpoints and views, the standard fosters a holistic understanding of the system, accommodating the various needs and perspectives of stakeholders. ISO/IEC/IEEE 42010, with its emphasis on traceability and systematic documentation, serves as a crucial foundation for effective architecture development and evaluation, ultimately contributing to better decision-making and the successful realization of complex systems, such as NEPHELE. A short description of the major ISO/IEC/IEEE 42010 components follows:

- **Stakeholders** are individuals or groups with vested interests in the system, such as developers, end-users, and regulatory bodies. Their perspectives and requirements shape the architecture.
- **Concerns** represent the key issues and interests of stakeholders, reflecting their needs, constraints, and goals.
- **Views** are abstractions of the system from the perspective of specific concerns. They offer distinct snapshots of the architecture, focusing on relevant aspects.
- **Viewpoints** are the specifications that define how to construct and interpret views. They encapsulate the conventions, notations, and modeling techniques used to represent the architecture in a way that aligns with stakeholders' concerns.

Together, stakeholders, concerns, views, and viewpoints form a structured framework within ISO/IEC/IEEE 42010, enabling effective communication and documentation of architectural decisions in a manner that is both comprehensive and accessible to diverse stakeholders.

In this deliverable, we present a detailed architecture description of the NEPHELE ecosystem based on the ISO/IEC/IEEE 42010 standard, through generating a structured and coherent description of the IoT Edge-to-Cloud architecture that facilitates effective communication, collaboration, decision-making and synergy throughout the development and deployment lifecycle. Through the integration of ISO/IEC/IEEE 42010 to the NEPHELE’s ecosystem, the detailed description, the interoperability of the fundamental components of the platform, the integration of the Functional Requirements (FR) and key design principles will be extracted. The architectural description of the major NEPHELE’s components is derived as the outcome.

In the NEPHELE context, we aim to describe three core architectural components: i) the Meta-orchestration Platform, ii) the VOStack open-source IoT and edge computing software stack and iii) the Dashboard and Development environment. In applying ISO/IEC/IEEE 42010 to describe the Meta-Orchestrator Platform, VOStack, and the Dashboard and Development environment within the NEPHELE ecosystem, we leverage a unified and standardized approach for articulating their architectures. The Meta-Orchestrator Platform, responsible for task scheduling, resource management,

monitoring, and application deployment, can be systematically described by defining stakeholders, concerns, views, and viewpoints. This enables a comprehensive understanding of its role, ensuring its maintenance, validation, and updates align with the diverse interests of system stakeholders. Similarly, the VOStack, designed to manage IoT devices through virtualized instances, benefits from ISO/IEC/IEEE 42010 by structuring its description around stakeholders, concerns, views, and viewpoints. This approach facilitates clear communication among stakeholders, validating and updating its functionalities effectively. The Dashboard and Development environment, serving as the user interface and collaborative workspace, can be characterized using ISO/IEC/IEEE 42010 to describe the underlying architecture, fostering effective collaboration and development. Overall, the application of ISO/IEC/IEEE 42010 to these components ensures a structured and coherent representation, promoting clear communication, collaboration, and informed decision-making throughout their lifecycle, ultimately contributing to the success of the NEPHELE ecosystem.

4. Stakeholders, Concerns, Views and Viewpoints

4.1. Stakeholders and High-Level Concerns

This section presents an overview of the different stakeholders involved in the NEPHELE's ecosystem from various perspectives and viewpoints. Each individual stakeholder has its own objectives, concerns and challenges. These aspects can be either common/overlapping, however they can also be contradicting, thus the special design choices are configured and the corresponding architectural components are fine-tuned, aiming to co-satisfy the different stakeholder entities. In the NEPHELE ecosystem, the relationship between stakeholders and entities follows a 1-to-N mapping, signifying that each stakeholder is associated with multiple entities within the ecosystem. In essence, the 1-to-N mapping signifies a complex web of relationships, where each stakeholder's influence spans across multiple entities, thus requiring collaboration and synergy across the NEPHELE ecosystem. This interconnected mapping is instrumental in ensuring that the diverse objectives and concerns of stakeholders are effectively addressed throughout the architecture, promoting a holistic approach to system design and operation.

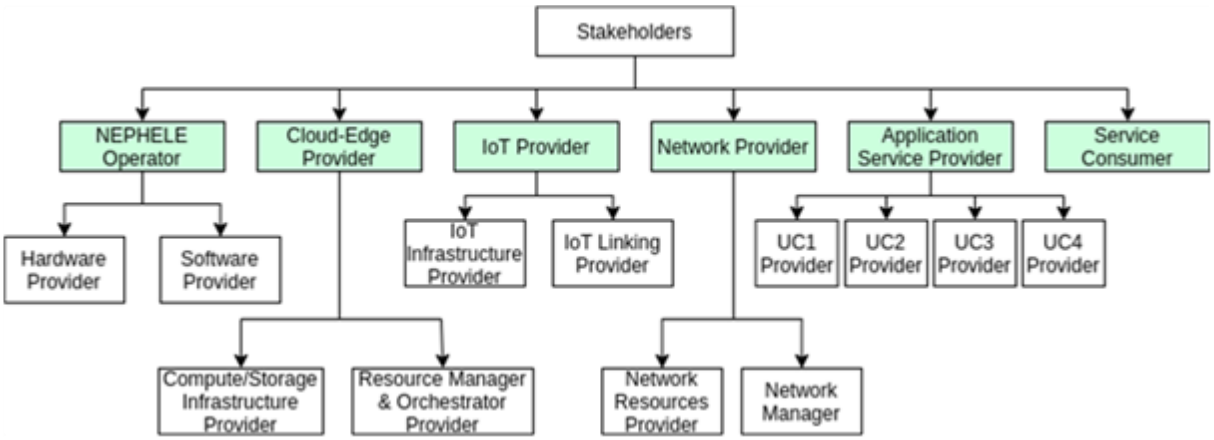

Figure 1. Overview of the NEPHELE Stakeholders.

The major stakeholders are depicted in Figure 1 and analyzed in the following text:

1. **NEPHELE Operator:** Corresponds to the owner of the NEPHELE platform from the perspective of the platform's management both in terms of hardware infrastructure and software utilities required to maintain the whole platform fully functional.
    a. **Objectives:**
        i. Ease of scalability of the underlying infrastructure to extend the coverage of the ecosystem while maintaining security.
        ii. High-Availability (HA) and resilience of the synergetic orchestration platform.
        iii. Lifecycle management of the Virtual Objects (VOs) (optional) and the HDA components.
        iv. Ease of usability and Use Case (UC)-oriented functionalities to promote the use of the platform.
        v. Efficient resource utilization of the compute and network resources for a cost-effective and efficient operation of the platform.
    b. **Concerns:**
        i. Cost-effective maintenance and scalability.

ii. Compliance with regulatory standards.
iii. Rapid response to hardware/software failures.
iv. Data backup and disaster recovery

2. **Cloud-Edge Provider (CEP):** This corresponds to the NEPHELE infrastructure provider in terms of computation and storage resources. The major responsibility of the CEP is the offering of resources and the effective management and orchestration of the HDAs deployed across the IoT-to-Edge-to-Cloud continuum. Furthermore, the hardware infrastructure of the IoT devices and end-servers are considered as part of the available end-to-end infrastructure. CEP integrates decisions within the orchestration actions, exploiting advances provided by several Artificial Intelligence (AI) technologies in features detection and inference and leading to the optimal management of the interplay among edge and cloud resources.
    a. **Objectives:**
        i. Efficiently manage and orchestrate Hyped Distributed Applications (HDA).
        ii. Optimize resource allocation across IoT, Edge, and Cloud.
        iii. Utilize AI technologies for resource management.
        iv. Ensure high availability and performance.
    b. **Concerns:**
        i. Security of computing and storage resources.
        ii. Minimizing resource wastage.
        iii. Interoperability between IoT hardware, VOs and the HDA software deployed within their infrastructure
        iv. Compatibility with various HDA configurations.
        v. End-to-end (E2E) isolation and multi-tenancy to support multiple applications at the same time.
        vi. Handling resource spikes and demand fluctuations.

3. **Internet of Things (IoT) Provider:** this entity provides access to IoT infrastructure that is already deployed and operational (e.g., IoT infrastructure that exists in smart cities). Such infrastructure can be used within the NEPHELE ecosystem through the provision of virtual counterparts of the IoT devices and their interlinking with edge/cloud computing applications.
    a. Objectives:
        i. Manage and provide access to deployed IoT infrastructure.
        ii. Introduce advanced services taking advantage of the NEPHELE ecosystem.
    b. Concerns:
        i. Conflict resolution in case of access by multiple tenants.
        ii. Secure access to the devices and the available data.
        iii. Consider data privacy aspects.
4. **Network Provider (NP):** The NP is the entity being responsible for the effective network operation of the NEPHELE platform. This corresponds to the management and maintenance of network resources across the continuum, aiming to provide a robust network infrastructure that can cover the various HDA needs, i.e. high-speed, security and Quality of Service (QoS). Moreover, the operation of NP strongly relies on the effective collaboration with the CEP, by integrating sophisticated mechanisms for compute-network synergy.

a. **Objectives:**
    i. Ensure robust network operation across the continuum.
    ii. Provide high-speed, secure, and QoS-compliant network services.
    iii. Collaborate effectively with CEP for compute-network synergy.
    iv. Monitor and maintain network resources proactively.
b. **Concerns:**
    i. Network congestion and latency.
    ii. Security breaches and data interception.
    iii. Adapting to dynamic HDA network requirements.
    iv. Network scalability and fault tolerance.
    v. Support multiple HDA and users at the same time

5. **Application Service Provider/Developer (ASP/ASD)**: These are the HDA developers and providers who offer an E2E application to service consumers. They provide the IoT with the pre-configuration required as well as the HDA components. They should provide the Application Graph to the NEPHELE Operator as well as any additional information required to correctly orchestrate and automate the lifecycle of the different parts of the HDA. This stakeholder might be further composed of two distinct actors, the Application Service Developer (ASD), which develops part or the totality of an HDA, and the Application Service Provider (ASP) which uses assets from the application developer and of their own to create a consumer - tailored service offering through the NEPHELE platform.
    a. **Objectives:**
        i. Develop and provide end-to-end applications to service consumers.
        ii. Assert that the Service Level Agreements (SLAs) are maintained to the service consumer.
        iii. Offer pre-configured IoT and HDA components.
        iv. Collaborate with the NEPHELE Operator for proper orchestration.
        v. Ensure efficient automation of HDA lifecycle.
        vi. Publish/Subscribe services in the NEPHELE ecosystem to extend their applications.
    b. **Concerns:**
        i. Application scalability and performance.
        ii. Compatibility with NEPHELE platform.
        iii. Protecting intellectual property and data.
        iv. Meeting SLAs and user expectations.

6. **Service Consumer**: They play a vital role in the NEPHELE ecosystem by serving as the end-users of the HDAs offered through the platform. Their primary role is to access and utilize these HDAs to meet their specific needs and objectives. Service consumers expect a seamless experience with high-quality services, minimal disruptions, and a user-friendly interface. They rely on the NEPHELE platform to deliver on these expectations, ensuring that the applications they use are not only accessible but also reliable and secure. Additionally, service consumers may have diverse requirements and preferences, and they often benefit from tailored service offerings provided by ASPs and Application Graphs that align with their unique use cases.

a. **Objectives:**
   i. Access and utilize HDAs from a variety of locations and devices.
   ii. Experience high-quality services with minimal disruptions even in mobility and high-service-traffic scenarios.
   iii. Meet their specific needs through tailored service offerings.

b. **Concerns:**
   i. Data privacy and security.
   ii. Service reliability and availability.
   iii. Ease of use and user experience.
   iv. Cost-effectiveness of services consumed.

4.2. Architecture Views and Viewpoints

In this section we present the overall architecture views and viewpoints. By organizing the representation into interconnected architecture views, we provide a holistic understanding of the framework, addressing various concerns and perspectives of the NEPHELE system. The viewpoints provide a comprehensive understanding of the architecture from different stakeholders' perspectives, allowing for effective communication, analysis, and decision-making throughout the system's lifecycle. For each individual viewpoint, we split its fundamental components on the three dominant components of the NEPHELE ecosystem: i) the Meta-Orchestration platform, ii) the VOStack and iii) the Dashboard and Development environment. In each case, upon the description of the viewpoint, we provide the requirements that are fulfilled based on the support of the challenges and functionalities identified in the viewpoint. The list of requirements is coming from their specification in D2.1 [11]. The various viewpoints are depicted in Figure 2 and analyzed in the following subsections.

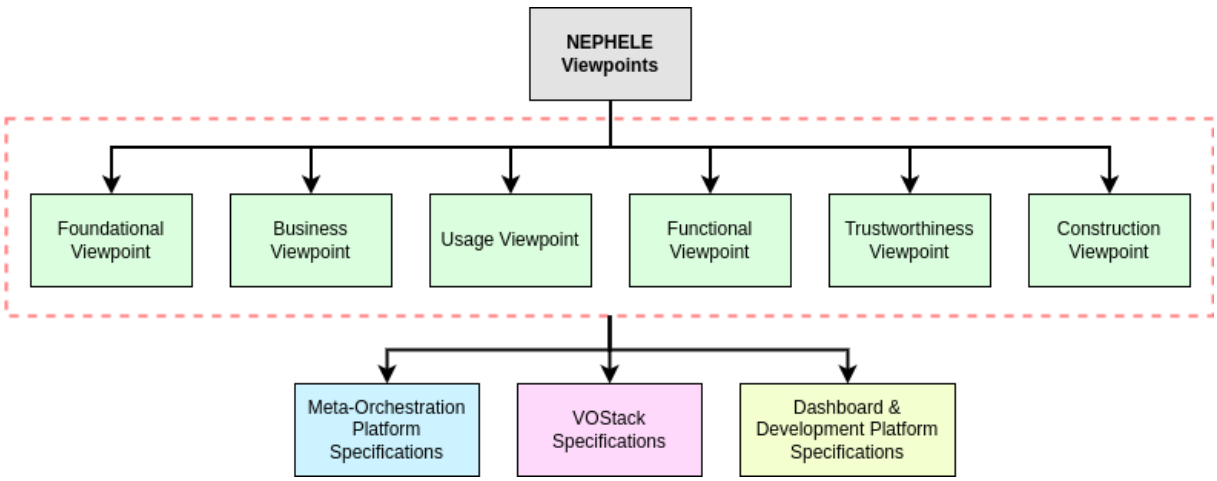

Figure 2. Overview of the NEPHELE Viewpoints and Specification Mapping

4.2.1. Foundational Viewpoint

The Foundational Viewpoint within the NEPHELE IoT-to-Edge-to-Cloud architecture defines the bases upon which the entire ecosystem is built. Its primary purpose is to define, establish, and maintain the essential foundational elements that ensure the platform's stability and reliability. This viewpoint addresses concerns related to hardware infrastructure, encompassing the physical devices and the end-

servers, as well as the software utilities required to manage and maintain these resources. The corresponding stakeholders in the foundational viewpoint aim to guarantee that the NEPHELE platform remains fully operational and capable of supporting a wide array of services and applications spanning from IoT to Edge to Cloud. Within the foundational viewpoint, stakeholders, including the NEPHELE Operator, CEP and NP make decisions about hardware and software configurations. These decisions not only facilitate seamless operation but also enable scalability, cost-effectiveness, adaptability, efficient resource allocation and orchestration. The foundational viewpoint serves as the major principle for the platform's growth, resilience, and capacity to meet the diverse objectives and concerns of the various stakeholders.

4.2.1.1 Meta-Orchestration Platform Specifications

Table 1. Meta-Orchestration platform specifications for the foundational viewpoint

Viewpoint Name	Foundational Viewpoint for the Meta-Orchestration Platform	
Overview	Considers the concerns related to the essential characteristics that have to be provided by the Meta-Orchestration Platform	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, IoT Provider
	Concerns	- How orchestration of applications can be done over resources in the computing continuum? - What level of distribution is possible and efficient among orchestration mechanisms? - How the interplay of different stakeholders (e.g., cloud provider, network provider) can be supported? - How can a meta-orchestration solution be applicable in different scenarios (generalization)? - How can the IoT infrastructure be considered as a manageable part of the continuum? - What kind of abstractions can be given to application developers to hide from them the complexity in terms of resources management? - How can we optimally satisfy the objectives of application providers and infrastructure providers during the provision of distributed applications?
	Model kinds/Legends	The foundational view/elements of the Meta-Orchestration platform include the following: - System of Systems approach - Intent-driven approach - Multi-cluster orchestration mechanisms - Interplay between Cloud/Edge and Network Providers - Virtualization of IoT devices and Convergence with Edge/Cloud Computing Technologies
Related requirements (from D2.1 [11]): FR_SO_001, FR_SO_002, FR_SO_004, FR_SO_009, FR_SO_012, FR_SO_014, FR_SO_017, FR_SO_020, FR_SO_021, FR_SO_022, NFR_SO_001,		

NFR_SO_007

System of Systems approach: A "system of systems" approach is introduced for managing the deployment of distributed applications across resources in the computing continuum. Based on the "system of systems" approach, a centralized entity is responsible for the deployment of a distributed application, while the control is distributed among several entities. A hierarchy may be introduced in decision-making, where each entity is able to undertake actions applicable to the environment that it can manage (set of lower-level entities that report to it).

Intent-based Orchestration: We are considering the development of an intent-driven orchestration approach, where a high-level intent description can be translated to deployment and operational policies, applicable across resources in the computing continuum. A main challenge regards the provision of descriptive models for the specification of intent that can be easily adopted by application providers and end users. Furthermore, there is a need for translation mechanisms that can decompose the intent into policies applicable over the available resources. The latter can be assisted through the adoption of machine learning techniques (e.g., Natural Language Processing techniques).

Multi-cluster orchestration mechanisms: Based on the transition from centralized to distributed continuum systems, there is evident a need to manage computing infrastructure that includes resources across the computing continuum from IoT to edge to cloud computing resources. Various clusters have to be managed by a centralized entity to support the deployment and lifecycle management of distributed applications. Various multi-cluster management mechanisms and (open-source) tools emerge, able to abstract resources toward a higher-level entity (within a hierarchy) that can manage them in a homogeneous way. We are considering the adoption of emerging open-source solutions for multi-cluster management and the development of hierarchical decision-making mechanisms based on them. AI-assisted orchestration mechanisms are introduced to improve automation and decentralized intelligence characteristics.

Interplay between Cloud/Edge and Network Providers: Interaction between providers of compute and network infrastructure is required in some cases for the efficient provision of distributed applications, especially when the applications have strict network and QoS requirements that have to be satisfied. In this case, the provision of interfaces to cloud/edge providers for the management of the network infrastructure is considered as an enabler for guaranteeing strict SLAs to end users. The current trend regards the development of open Application Programming Interfaces (APIs), where network functionalities can be activated and managed on demand by edge/cloud computing platforms. In NEPHELE, we consider such types of interactions and specification of open APIs.

Virtualization of IoT devices and Convergence with Edge/Cloud Computing Technologies: A key challenge lies in the convergence of IoT technologies with the computing continuum and the seamless interaction of diverse IoT devices and distributed applications. This encompasses integrating IoT operations with edge and cloud computing, demanding transparent deployment and orchestration for applications across the computing continuum [7]. Virtualization of IoT devices is considered beneficial for solving IoT interoperability aspects, providing advanced capabilities to IoT applications based on the combination of resources at the IoT and edge part of the infrastructure, as well as advanced security, privacy and network management mechanisms. Through virtualized functions, convergence of IoT functions with edge/cloud computing applications is also realistic, based on the specification of open interfaces and interaction schemes. Moreover, the enhancement of resource constrained devices with advanced functionalities is necessary to facilitate the deployment of application graphs that interact with physical devices. Besides, the development of a virtual counterpart of a physical device is crucial in solving multi-tenancy issues and increasing security towards the applications. Tackling challenges related to IoT interoperability and convergence with edge/cloud computing technologies is a core part

of NEPHELE, where the focus is given on the development of an open-source software stack for this purpose.

4.2.1.2 Virtual Object Stack (VOStack) Specifications

Table 2. VOStack specifications for the foundational viewpoint

Viewpoint Name	Foundational Viewpoint for the VOStack	
Overview	Considers the concerns related to the essential characteristics that have to be provided by the VOStack	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, IoT Provider
	Concerns	- What are the implications of the concept of the Virtual Object in terms of management of IoT infrastructure and interaction with edge/cloud computing applications? - How can heterogeneity in terms of communication and semantic protocols be tackled? There is a need for open and interoperable solutions. - How can convergence aspects with edge/cloud computing applications be tackled? The VOs should be interoperable with application graphs. - The concept of Digital Twin (DT) has to be accommodated within the design of VOStack. - What is the main added value introduced by VOStack? - How can VOStack be used for innovative new business models?
	Model kinds/Legends	The foundational view/elements of VOStack includes the following: - Definition of VOs and cVOs and specification of their role - Conceptualization of the main VOStack Layers and the functionalities per layer
Related requirements (from D2.1 [11]): FR_VOS_001, FR_VOS_002, FR_VOS_005, FR_VOS_007, FR_VOS_008, NFR_VOS_03, NFR_VOS_04, NFR_VOS_05,		

Following, we provide a short description of the main concepts related to the VOStack. Detailed description of these concepts is provided in D2.1.

Definition of VOs and cVOs and specification of their role:

A **Virtual Object (VO)** is considered as a virtual counterpart of a physical device on the Internet of Things domain. It provides a set of abstractions for managing any type of IoT device through a virtualized instance while augmenting the supported functionalities through the development of a multi-layer software stack, called Virtual Object Stack (VOStack) [11].

A **Composite Virtual Object (cVO)** is a software entity that can manage the information coming from one or multiple VOs and provide advanced functionalities. The cVO interacts with the VOs, processes the collected information, and can contextually produce advanced knowledge, by enabling the communication and collaboration of several VOs, toward the production and exposure of a combined

set of data outputs [11]. Under the scope of this mode, the cVO can operate also as a Digital Twin (DT). In this case, the enhanced capabilities of the VO through the cVO refer to the support of simulation, integration, testing, monitoring, and maintenance activities for an IoT device [11].

Conceptualization of the main VOStack Layers and the functionalities per layer:

The development of an IoT and edge computing software stack in the NEPHELE project is motivated by the need of supporting a full convergence and integration among existing and evolving IoT and edge computing technologies. A software stack (VOStack) is under development to flexibly support interaction with both physical IoT devices and edge/cloud computing orchestration platforms, considering both the VO and cVO concept. The main incentive is that the VOs must be lightweight and modular while supporting basic functionalities that most devices and applications need. Hence, the VOStack has three main architectural layers namely: (i) the Physical Convergence Layer, (ii) the Edge/Cloud Convergence Layer, and (iii) the Backend Logic Layer. Detailed description of the VOStack layers is provided in D2.1 [11].

4.2.1.3 Dashboard and Development Environment Specifications

Table 3. Dashboard and Development environment specifications for the foundational viewpoint

Viewpoint Name	Foundational Viewpoint for the Dashboard and Development Environment	
Overview	Considers the concerns related to the essential characteristics that have to be provided by the NEPHELE Dashboard and Development Environment.	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, Application Service Provider/Developer, CEP
	Concerns	- How can the various artifacts offered in the Registry be modeled to be easily exploitable by the various stakeholders? - How do we manage ownership/copyright aspects? - What kind of discovery options are going to be made available for the offered artifacts? - How validation of artifacts can be supported to guarantee compliance with the NEPHELE specifications? - Provide real-time information to CEP for the lifecycle management of HDAs. - The design of a dashboard that can be targeted to various stakeholders is challenging.
	Model kinds/Legends	- Basic objectives and high-level characteristics of the Development Environment - Basic objectives and high-level characteristics of the Dashboard
Related requirements (from D2.1 [11]): FR_VOS_007, FR_SO_011, FR_SO_012, FR_SO_013, FR_SO_014, FR_SO_016, FR_SO_017, FR_SO_018, FR_SO_019, NFR_SO_005		

Basic objectives and high-level characteristics of the Development Environment: The implementation of a Development Environment that can act as a central repository for developing, storing and making available the produced software artifacts is required. The development environment has to be targeted to developers. Functionalities for software development, editing and validation of descriptors, software storage and availability have to be supported. In this way, the produced software can be easily hosted, re-used, and/or extended. Validation of software artifacts is supported through a

set of mechanisms that examine the produced software descriptors. Authentication mechanisms will be in place to provide access to the development environment, based on the profile of each user.

Basic objectives and high-level characteristics of the Dashboard: The NEPHELE dashboard provides access through an intuitive user interface to the various NEPHELE stakeholders, focusing on the application and infrastructure providers. It provides views for interacting with the development environment, getting access to the various software artifacts, monitoring the available infrastructure across the computing continuum, and monitoring the lifecycle of the deployment of distributed applications over the available infrastructure.

4.2.2. Business Viewpoint

The business viewpoint in the context of the NEPHELE system focuses on aligning the platform's strategic and economic goals with its operational capabilities. The major objective is to optimize the performance of the ecosystem from a business perspective, ensuring that it not only meets technical requirements but also delivers value to stakeholders and maintains its competitive nature. Key stakeholders, including the NEPHELE Operator, CEP, ASPs, and Service Consumers collaborate within the business viewpoint to make informed decisions about resource allocation, pricing models, and market positioning. Furthermore, the business viewpoint enables stakeholders to identify potential growth opportunities and adapt the platform's offerings to evolving market dynamics. The business viewpoint ensures that the NEPHELE platform thrives in the competitive landscape, fostering long-term success and viability in the IoT, Edge, and Cloud ecosystem.

4.2.2.1 Meta-Orchestration Platform Specifications

Table 4. Meta-Orchestration platform specifications for the business viewpoint

Viewpoint Name	Business Viewpoint for the Meta-Orchestration Platform	
Overview	Considers the concerns to be addressed by the business views of the Meta-Orchestration Platform	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, IoT Provider, ASPs, Service Consumers
	Concerns	- Management of highly distributed applications over distributed resources in the continuum needs the development of business models where reservation of resources may take place from different providers across the continuum. Resources reservation may be dynamic based on the workloads and associated with relevant billing models. Usage analytics can be made available to facilitate this reporting and billing. - Collaboration between cloud/edge computing providers and network providers is required in case of deployments with strict QoS (e.g., latency sensitive) or security needs (e.g., network isolation). - Synergies among cloud/edge computing providers, network providers, IoT providers and the NEPHELE operator have also to be established to take advantage of the provided solution.

		- Development of an open-source and modular software stack based on open and interoperable APIs is required given the complexity and high heterogeneity of deployment scenarios that can be considered across the continuum. Customized solutions can be developed for specific needs. A one size fit all solution cannot be applicable to a wide range of scenarios. - Specific billing agreements must be designed for the coexistence of virtual deployments from different stakeholders on top of shared hardware resources
	Model kinds/Legends	- Business relevance of a Meta-Orchestration System - Need for an open Continuum Stack
Related requirements (from D2.1 [11]): FR_SO_015, FR_SO_017, NFR_SO_001		

Business relevance of a Meta-Orchestration System: The design of the NEPHELE Meta-Orchestration system is based by design on openness, modularity and interoperability principles. It is designed based on open-source solutions, while open APIs are going to be specified for the interaction among the various components. A set of APIs concern interaction among components managed by different stakeholders, considering the need for interoperability between the Meta-Orchestration System and the VOStack, as well as the interoperability between compute and network resources management. In this way, the produced system can be adopted and extended by the various targeted business stakeholders and facilitate the management of distributed applications over the computing continuum. The adoption of intent-driven orchestration approaches provides the ability to service consumers and application providers to agree on high level goals to be achieved during deployment and associate them with billing options.

Need for an open Continuum Stack: The NEPHELE Meta-Orchestration system aims to contribute towards the development of an open Continuum Stack, based on the provision of open and modular components that can be adopted by various stakeholders and platforms. Business opportunities may be associated with the overall meta-orchestration system as an integrated environment, as well as for specific components, including the AI-assisted orchestration mechanisms, multi-cluster management mechanisms and network management mechanisms.

4.2.2.2 Virtual Object Stack (VOStack) Specifications

Table 5. VOStack specifications for the business viewpoint

Viewpoint Name	Business Viewpoint for the VOStack	
Overview	Considers the concerns to be addressed by the business views of VOStack	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, IoT Provider, ASPs, Service Consumers
	Concerns	- How can VOStack be adopted and used in vertical industries? - How can the (c)VOs be used for innovative new business models?

		- Can IoT technologies interoperability and convergence facilitate business adoption of VOStack?
	Model kinds/Legends	- Business relevance of the VOStack - Alignment with standardization working groups - IoT Devices interoperability
Related requirements (from D2.1 [11]): FR_VOS_001, FR_VOS_002, FR_VOS_003, FR_VOS_005, FR_VOS_007		

Business relevance of the VOStack: VOStack regards an open-source software stack that supports virtualization of IoT devices and functions. It tackles challenges related to IoT protocols interoperability and IoT technologies convergence with edge and cloud computing technologies. VOStack has high business value since it can be adopted, customized and or extended to cover business needs in the IoT domain. VOStack can also be adopted to develop Digital Twins that can interact through the VOs with the available IoT infrastructure.

Alignment with standardization working groups: The design and development of VOStack is based on the specifications provided by emerging standardization bodies and working groups, while interoperability among different specifications is also targeted. The main working groups and specifications regard the W3C Web of Things working group and the Open Mobile Alliance (OMA) Lightweight M2M (LwM2M) specifications.

IoT devices Interoperability: Another crucial aspect is the interoperability of IoT devices. The landscape, saturated with manufacturers crafting devices embedded with proprietary systems complicates the pursuit of seamless integration. Also, IoT devices often use different communication protocols and standards, and generate vast amounts of data in various formats making it difficult for them to seamlessly interact with each other. Efforts toward standardization, encompassing IoT protocols, APIs, and data formats, aspire to bridge these chasms.

4.2.2.3 Dashboard and Development Environment Specifications

Table 6. Dashboard and Development environment specifications for the business viewpoint

Viewpoint Name	Business Viewpoint for the Dashboard and Development Environment	
Overview	Considers the concerns to be addressed by the business views of the Dashboard and Development Environment	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, Application Service Provider/Developer, Service Consumers, IoT Provider
	Concerns	- Need to define business models between NEPHELE Operator and ASPs for storing, managing and giving visibility to each of the artifacts stored. - Need to define business models between NEPHELE Operator and Service customers to have access to the catalog of artifacts. - Need to define business models between NEPHELE Operator, ASPs and Service customers for artifacts download/usage. - Need to support pricing models for each of the artifacts stored (i.e., freemium models).

		- Need to define the Intellectual Property Rights (IPR) strategy to be followed for each of the artifacts stored. - Need to support a versioning strategy for each of the artifacts stored. - How to grant cost-efficient storage; How to ensure scalability of storage - Need to provide a user-friendly interface - How to ensure that the search engine of artifacts is easy to use and effective - Need to comply with current standards and regulations - Flexibility to adapt to new market trends, standards and regulations - Definition of a strong value proposition to be competitive in the market required; How to provide added value to ASPs and - Service Consumers to differentiate in the market - Creation of a ecosystem ASPs/ Service Consumers may be required to grant success - Possibility of validating artifacts without uploading them (IPR concerns)
	Model kinds/Legends	- Development Environment - Dashboard
Related requirements (from D2.1 [11]): FR_SO_012, FR_SO_016, FR_SO_019		

Development Environment: The NEPHELE development environment provides an open-source environment to application developers to develop, store, and make available the software artifacts that compose the distributed applications. It is based on open technologies and APIs, making it easily adaptable and extensible by interested parties. The NEPHELE development environment aims to support efficient storage and retrieval of deployment artifacts, optimizing costs related to data storage. It should facilitate rapid deployment of updates and changes to align with the evolving needs of the business. It is interconnected with the NEPHELE Dashboard where graphical access to the developed artifacts is provided. It is envisaged to support different roles for introducing and updating the available software, considering the different stakeholders that may have access to the development environment and the public or private availability of the produced software artifacts.

Dashboard: The NEPHELE Dashboard is targeted mainly to the NEPHELE operator, who is responsible for managing the operation of the NEPHELE platform and enabling the deployment of distributed applications across resources in the computing continuum. A set of user-friendly interfaces significantly reduces the overall administration overhead for the management of the developed software and the various deployments. Customized views are also targeted to other stakeholders, including CEP, IoT providers, application service providers, and application developers. The Dashboard is going to consume the information provided through open APIs by the Development Environment and the Synergetic Meta-Orchestrator (SMO).

4.2.3. Usage Viewpoint

The usage viewpoint focuses on the practical experience of end-users who interact with the NEPHELE platform. Its primary purpose is to ensure that the various components are accessible, user-friendly, customizable, and capable of satisfying the end users. Within this viewpoint, concerns such as ease of use, user experience design, tailoring services to individual preferences, and accommodating various service requirements come to the forefront. Stakeholders in the usage viewpoint collaborate to create an environment where end-users can access and derive value from the platform's services, ultimately driving user satisfaction and adoption. This viewpoint addresses the needs of all the NEPHELE stakeholders.

4.2.3.1 Meta-Orchestration Platform Specifications

Table 7. Meta-orchestration platform specifications for the usage viewpoint

Viewpoint Name	Usage Viewpoint for the Meta-Orchestration Platform	
Overview	Considers the concerns to be addressed by the usage of the Meta-Orchestration Platform	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, IoT Provider, ASPs
	Concerns	- Ease registration of computational infrastructure across the computing continuum. - Support interaction between computing and network providers to provide network management mechanisms to assure the desired QoS level. - Lifecycle management of the deployment of application graphs. - Hierarchical approach in the management of resources in the various clusters. - Specification and management of intent-driven deployments, considering the SLAs that are agreed between the application providers and the service consumers. - High level description of the intent and support of translation mechanisms to policies. - Monitoring of deployment status and application metrics. - The time to provision and deploy an application shall be controlled and minimized. - The time required to react over an alarm for a specific deployment shall be controlled and do not impact the overall application performance.
	Model kinds/Legends	- Openness - Modularity - Interoperability Most of the functionalities related to the usage viewpoint are supported by the NEPHELE Dashboard, as detailed in subsection 4.2.3.3.

Related requirements (from D2.1 [11]): NFR_SO_001, NFR_SO_002, NFR_SO_007, NFR_SO_009

Openness, modularity and interoperability are principles considered by design in the development of the Meta-orchestration platform to enable its adoption, usage and extension by interested parties.

4.2.3.2 Virtual Object Stack (VOStack) Specifications

Table 8. VOStack specifications for the usage viewpoint

Viewpoint Name	Usage Viewpoint for the VOStack	
Overview	Considers the concerns to be addressed by the usage of the VOStack	
Viewpoint specification	**Known typical stakeholders**	IoT Provider
	Concerns	- Ease development of VO and cVO descriptors based on the provided templates - Ease adoption and extension of the software stack to introduce IoT virtual functions for VOs and cVOs - Provision of efficient access to the available IoT data - Onboarding of analysis processes that can be executed over the available IoT data - Ease integration of VOs and cVOs into application graphs. - Ability to develop Digital Twins and support operation in emulation and simulation mode. - Monitoring of the status of the VOs/cVOs (e.g., health checks) - Semantic interoperability of the collected data based on specifications such as W3C WoT, NGSI-LD, OMA LwM2M
	Model kinds/Legends	- Openness - Modularity - Interoperability Most of the functionalities related to the usage viewpoint are supported by the NEPHELE Dashboard, as detailed in subsection 4.2.3.3.
Related requirements (from D2.1 [11]): NFR_VOS_03, NFR_VOS_04		

Openness, modularity and interoperability are principles considered by design in the development of the VOStack to enable its adoption, usage and extension by interested parties.

4.2.3.3 Dashboard and Development Environment Specifications

Table 9. Dashboard and Development environment specifications for the usage viewpoint

Viewpoint Name	Usage Viewpoint for the Dashboard and the Development Environment
Overview	The usage viewpoint is concerned with how end-users interact with and utilize the HDAs. It addresses concerns such as ease of use, user experience, customization

	options, and specific service requirements. This viewpoint ensures that the platform meets the practical needs of consumers.	
Viewpoint specification	**Known typical stakeholders**	CEP, ASP, ASD, Service Consumers
	Concerns	- How quick and centralized it is to access the resources for learning? - How big is the learning curve of the NEPHELE's specific artifacts and how similar it is to other technologies known? - Is it possible to reuse or connect to other proprietary applications running outside of the NEPHELE's framework? - The HDA Registry (HDAR) should give the option to have public and private repositories for each artifact. - The HDAR should allow users to quickly amend artifacts uploaded and keep track of different versions in the same way as other technologies such as Helm or Docker. - Application developers need to have a way to programmatically interact with the HDAR. - ASPs need to have a dashboard where they could obtain information about their artifacts before and after instantiation. - There should be a single user authentication and authorization mechanism to interact with the Dashboard and all potential sources of graphical information (HDAR, SMO platform, metrics, logs, events, etc.) and, if possible, with the Development Repository and documentation. - The Dashboard should be designed as simple and clear as possible to be user-friendly. - Dashboard, HDAR and SMO platforms should be synchronized and should alert of actions that could lead to a faulty state of the HDA such as deleting an artifact that is being used in a high mobility scenario. - Monitoring of the status of the application components, the resource usage and the overall application metrics during each deployment.
	Model kinds/Legends	- Openness - Modularity - Interoperability - UI/UX Effectiveness and Usability - Effective Data Visualization - Collaborative Development Tools Integration
Related requirements (from D2.1 [11]): NFR_VOS_04, NFR_VOS_07, NFR_VOS_10, FR_SO_015, NFR_SO_004, NFR_SO_009		

Openness, modularity and interoperability are principles considered by design in the development of the dashboard and the development environment to enable its adoption, usage and extension by interested parties.

User Interface / User Experience (UI/UX) Effectiveness and Usability: The effectiveness and usability of the UI and user UX in the Dashboard and Development Environment pose significant challenges within the NEPHELE ecosystem. Poor UI/UX design can lead to issues such as increased cognitive load, inefficient navigation, and a lack of clarity, hindering the overall usability of the platform. Users may face difficulties in comprehending complex architectural information, collaborating on development tasks, or accessing critical functionalities, impacting their efficiency and satisfaction. A well-designed UI/UX, on the other hand, can offer a host of benefits. Clarity in presentation and intuitive navigation enhance the overall usability, allowing users to quickly handle architectural details and seamlessly navigate the development environment. Effective data visualization techniques within the dashboard ensure that performance metrics, system status, and architectural information are presented in a comprehensible manner, aiding developers and stakeholders in making informed decisions. Enhanced UX promotes positive interaction, reducing the learning curve for users and increasing overall productivity. In essence, a good UI/UX design not only mitigates usability challenges but also contributes to a more efficient, satisfying, and collaborative user experience within the NEPHELE Dashboard and Development Environment.

Effective Data Visualization: is a crucial aspect of the Dashboard and Development Environment in the NEPHELE ecosystem, serving as a cornerstone for comprehending complex architectural information and making informed decisions. Inefficient data visualization may lead to misinterpretation, inefficient analysis, and reduced clarity. Inadequate visualization may obscure key trends, hinder the identification of performance anomalies, and degrade the overall understanding of the system's status. On the other side, a well-implemented data visualization strategy offers numerous advantages. It provides a clear representation of system metrics, allowing stakeholders to quickly identify patterns, trends, and potential issues. Intuitive visualizations enhance the accessibility of critical information, facilitating effective communication and collaboration among developers and other stakeholders. Interactive visual elements can enable users to drill down into specific details, offering a comprehensive view of the system's performance. Overall, effective data visualization not only addresses potential challenges but also enhances the interpretability of complex data, fostering a more insightful and actionable understanding of the NEPHELE ecosystem.

Collaborative Development Tools Integration: integrating collaborative development tools into the Dashboard and Development Environment of the NEPHELE ecosystem presents a critical challenge and opportunity. Ineffective integration may lead to fragmentation in workflows, hindering seamless collaboration among developers and impeding version control. Issues such as data inconsistencies, communication gaps, and a lack of real-time collaboration features can compromise the efficiency and cohesiveness of the development process. Successful integration of collaborative tools streamlines communication, ensures version control synchronization, and facilitates concurrent development efforts. A well-integrated environment empowers developers to work cohesively, share code seamlessly, and leverage real-time collaboration features, ultimately enhancing productivity and supporting the collaborative nature of software development within the NEPHELE platform.

4.2.4. Functional Viewpoint

Its purpose is to ensure that the NEPHELE ecosystem functions effectively and efficiently, meeting the diverse requirements and objectives of its stakeholders. It addresses concerns about managing HDAs, resource orchestration, network operations, and automation. By carefully defining and refining these functional aspects, stakeholders aim to create a robust and adaptable platform that seamlessly operates across the IoT-to-Edge-to-Cloud continuum. Stakeholders involved in the functional viewpoint include the NEPHELE Operator, CEP, NP, and ASPs.

4.2.4.1 Meta-Orchestration Platform Specifications

Table 10. Meta-orchestration platform specifications for the functional viewpoint

Viewpoint Name	Functional Viewpoint for the Meta-Orchestration Platform	
Overview	Considers the functionalities that have to be supported by the Meta-Orchestration Platform	
Viewpoint specification	**Known typical stakeholders**	CEP, NEPHELE Operator, Network Provider
	Concerns	- Intent-driven orchestration mechanism where a high-level intent can be translated to deployment and runtime re-configuration policies. - Support multi-objective optimization mechanisms for the deployment of distributed applications over resources in the computing continuum. - Support management of multi-cluster infrastructure based on resources' abstraction and unified management. - Support resource management actions such as horizontal scaling and compute offloading. - Increase intelligence and automation based on AI-assisted orchestration mechanisms. - Management of network resources and provision of network infrastructure (e.g., network slice) to address the network requirements of a distributed application - Support real time monitoring and data fusion at local (cluster) and global (end to end infrastructure) level - Manage different types of workflows (e.g., targeted to ML processes or serverless applications) - Support parallel deployments over the same infrastructure by different verticals - Support deployments of application graphs with integrated VOs and/or cVOs - Open programmable interfaces for interaction among the orchestration components
	Model kinds/Legends	- Openness - Modularity - Interoperability - Orchestration functions
Related requirements (from D2.1 [11]): FR_SO_001, FR_SO_002, FR_SO_003, FR_SO_004, FR_SO_005, FR_SO_007, FR_SO_008, FR_SO_009, FR_SO_010, FR_SO_011, FR_SO_012, FR_SO_013, FR_SO_014, FR_SO_015, FR_SO_016, FR_SO_017, FR_SO_020, FR_SO_023		

A wide set of functionalities is envisaged to be offered by the Meta-Orchestration platform. An elaborated description of the functionalities is provided in Section 5 of the document. Following, we provide some details for the main mechanisms that are considered.

Resource Orchestration during the deployment of Application Graphs: Resource Orchestration within the compute continuum involves the coordination and management of diverse resources across multiple domains, ensuring the efficient, reliable, and scalable operation of applications. Thus, the deployment of distributed application graphs across multiple domains in the continuum, emerges multifaceted challenges. Another significant challenge revolves around the optimal provisioning and orchestration of applications' distributed service nodes. This involves dynamically allocating resources based on varying demand patterns, to meet stringent QoS requirements. Efficient resource scheduling emerges as an uppermost concern in multi-domain orchestration [1]. This entails the optimal mapping of application service functions across distributed and heterogeneous resources while minimizing deployment cost and end-to-end latency. The challenge lies in orchestrating resource allocation in a manner that heterogeneous infrastructure or service providers and in some cases conflicting applications' requirements are met by optimizing applications overall performance and resource utilization [2]. Many studies on orchestration focus on resource allocation solutions considering emerging computing paradigms, such as Cloud Computing, Fog, and Mobile Edge Computing (MEC) [3]. The majority of them target a specific level of orchestration, and solve the corresponding optimization problems, such as Virtual Machine placement and migration [4] and resource scaling [5]. Also, integrating intent-driven orchestration mechanisms becomes essential, as they play a pivotal role in identifying the appropriate SLAs for each application, ensuring that the latter are provisioned in line with their specific requirements [6].

Task Scheduling: involves developing efficient task scheduling algorithms to optimize resource utilization and ensure timely execution, particularly in the context of dynamic workloads and fluctuating demands. Effective task scheduling is critical for optimizing resource usage and improving system performance by distributing tasks efficiently. Inadequate scheduling may lead to resource imbalances, increased latency, and poor throughput, impacting the overall reliability and scalability of the system. Effective task scheduling is essential for achieving optimal resource utilization, minimizing delays, and ensuring the system can efficiently handle varying workloads.

Real-Time Monitoring: This process involves the instantaneous collection, analysis, and visualization of key metrics and events, offering stakeholders immediate insights into the system's behavior. Real-time monitoring serves as a proactive mechanism, allowing administrators to maintain a proactive stance against emerging challenges. It facilitates the detection of irregularities in resource utilization, application performance, and network behavior, empowering timely interventions to prevent system degradation or failure. Additionally, real-time monitoring supports the adherence to SLAs. In the NEPHELE context, both inter-cluster and intra-cluster resource monitoring is dominant for the effective operation of the platform.

Seamless HDA Deployment: Seamless HDA deployment is dominant in the NEPHELE ecosystem due to its significance in ensuring efficient and error-free integration of applications across the IoT-to-Edge-to-Cloud continuum. The importance lies in the ability to effortlessly deploy applications, allowing for rapid adaptation to changing requirements and dynamic workloads. This flexibility facilitates quick responses to evolving user needs and market demands, contributing to the overall responsiveness and competitiveness of the NEPHELE platform. Issues triggered by non-seamless HDA deployment can significantly impact the system's functionality and performance. Delays or errors in deployment may result in increased latency, hindering real-time applications and compromising user experience. Inconsistent deployment across diverse hardware and software environments can lead to compatibility issues, affecting the interoperability of applications and potentially causing system failures. Furthermore, non-seamless deployment may introduce security vulnerabilities, exposing the system to potential breaches and compromising the integrity of data and sensitive information. Therefore, achieving seamless HDA deployment is crucial for maintaining the reliability, adaptability, and security of the NEPHELE ecosystem.

Network Orchestration: Network orchestration involves managing and coordinating the networking aspects within the compute continuum, including network connectivity, configuration, and security. Challenges in network orchestration include establishing and maintaining reliable network connectivity across multiple cloud environments, ensuring consistent network policies and security mechanisms, and optimizing network performance. Network orchestration also encompasses tasks such as load balancing, traffic optimization, and network function virtualization to achieve efficient and secure communication between applications and resources.

4.2.4.2 Virtual Object Stack (VOStack) Specifications

Table 11. VOStack specifications for the functional viewpoint

Viewpoint Name	Functional Viewpoint for the VOStack	
Overview	Considers the functionalities that have to be supported by the the VOStack	
Viewpoint specification	**Known typical stakeholders**	IoT provider, Application Developer, CEP
	Concerns	- Support the communication among the VOs and the IoT devices based on various communication protocols (e.g., HTTP, MQTT, CoAP) - Support semantic interoperability based on different semantic models - Connect heterogeneous IoT devices directly or through an IoT gateway - Enable the development of VOs and cVOs and their interlinking - Support storage and analysis of IoT data in internal storage spaces - Support the development of virtual functions that can be generic or device-specific - Support lifecycle management and health check of VOs - Support time sensitive networking functionalities for the interaction among IoT devices and edge/cloud infrastructure - The VO stack shall support time sensitive networking (TSN) functionalities for low-latency communication - The VO Stack shall enable the population of TSN schedule configurations into TSN bridges using a technology specific SouthBoundAPI (e.g., NETCONF/RESTCONF). - Support interfaces for the edge/cloud orchestration of VOs and cVOs - Support the interplay for the management of functions execution in the IoT and edge part of the infrastructure
	Model kinds/Legends	- Openness - Modularity - Interoperability - Edge/Cloud convergence

		- Backend logic (generic IoT functionalities) - Physical convergence
Related requirements (from D2.1 [11]): FR_VOS_001, FR_VOS_002, FR_VOS_003, FR_VOS_004, FR_VOS_006, FR_VOS_007, FR_VOS_008, FR_VOS_010		

A wide set of functionalities is envisaged to be offered by the VOStack. Such functionalities are grouped in three layers, namely the Edge/Cloud convergence layer, the Backend logic layer, and the Physical convergence layer.

Edge/Cloud convergence layer: This layer is responsible for bringing the VO closer to the application and orchestration layer. As the VO is part of the application graph, it communicates with entities, such as data consumers, applications, or users, through suitable interfaces. Various communication protocols (e.g., HTTP, MQTT, CoAP, etc.) are going to be supported. Hence, via this layer, IoT devices are exposed and consumed. More specifically, a set of functionalities may be provided through this layer for managing incoming requests and providing responses (e.g., requesting data, triggering actions, declaring new alerts), and handling multi-tenancy aspects (e.g., multiple requests for IoT device information). Besides, this layer supports a set of functions related to orchestration and addressing the monitoring of the status of the VO (e.g., container monitoring), the management of the deployment of the VO over the computing infrastructure (e.g., start, stop, restart, destroy), and the management of elasticity and migration actions.

Backend logic layer: This layer is responsible for augmenting the functionalities and capabilities of IoT devices. Here, we include all the logic related to the IoT device's operational behaviors, enhanced functionalities, and services that the Object/Device can perform. Primarily, the VOs are able to declare alerts on the IoT devices' state (e.g., a device suddenly restarted), and/or data-driven notifications (e.g., the temperature of a sensor rapidly increased). This functionality is closely related to the interaction with the storage entity, since, for instance, it is typical in many scenarios to observe past data values. Naturally, in trying to implement the virtual counterpart of an IoT device it is mandatory to introduce a set of actions and behaviors that the VO can dictate to the IoT device. To this extent, a VO can reconfigure or try to remotely heal a device. Moreover, following an event-based logic, actions are also either triggered by the monitored data (e.g., alerts and notifications coming from a sensor) or activated by commands received from the application and/or orchestration side (e.g., an application provider may want to dictate a different behavior of a sensor, such as changing the polling period of measurement when a given threshold is exceeded). Finally, for each defined action, a mechanism is designed to support the action-related policies implementing multi-tenancy characteristics. It is crucial that the set of actions, alerts, and notifications are reconfigurable and their definitions are not limited or heavily depend on the respective use case.

Physical convergence layer: This layer is responsible to tackle the major challenges of connecting the IoT devices with the computing continuum infrastructure. First and foremost, the VO is able to address device registration issues (e.g., registering a new device to a VO, bootstrapping of a connection, etc.). Regarding connectivity, a VO supports different types of communication protocols among those most widely used in the IoT domain at: (i) the application layer (e.g., MQTT, CoAP, HTTP, etc.), (ii) the network layer (e.g., IPv4, IPv6, etc.), and (iii) transport layer (e.g., TCP, UDP, etc.). In such a way, the majority of IoT devices are able to be connected and communicate with their virtual counterpart. However, as many devices have restricted security capabilities, authentication and authorization functionalities (e.g., OAuth 2.0) are provided to solve secure communications between the devices and the applications. Moreover, this layer simplifies the coordination of multiple IoT devices or IoT clusters,

by providing autonomic and self-* functionalities. Furthermore, a set of network-oriented functionalities are available to facilitate intermittent connectivity of the devices, manage dynamic routing protocols, time-sensitive networking mechanisms or tackle mobility aspects. On the one hand, by keeping the VO synced with the IoT device, clients are able to access the device's information uninterruptedly even if the device suddenly loses connection with the VO.

4.2.4.3 Dashboard and Development Environment Specifications

Table 12. Dashboard and Development environment specifications for the functional viewpoint

Viewpoint Name	Functional Viewpoint for the Dashboard and the Development Environment	
Overview	The functional viewpoint focuses on the features and capabilities of the NEPHELE platform. It deals with concerns related to HDA management, resource orchestration, network operation, and automation. Stakeholders in this view work to ensure that the platform functions effectively and efficiently.	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, ASP, ASD
	Concerns	- Support the development of distributed applications considering the produced software and the associated descriptors - Validate the proper specification of descriptors based on their templates - Store and make available the software artifacts to the application developers and providers (meta-orchestration platform) - Provide in the Dashboard views for managing the available software, the registered infrastructure, the deployment and lifecycle management of distributed applications - The Dashboard should offer a UI to show the metrics of the overall NEPHELE platform and the user's HDAs instantiated.
	Model kinds/Legends	- Openness - Modularity - Interoperability
Related requirements (from D2.1 [11]): FR_SO_010, FR_SO_012, FR_SO_014, FR_SO_019		

A wide set of functionalities is envisaged to be offered by the Dashboard and Development environment. An elaborated description of the functionalities is provided in Section 5 of the document.

4.2.5. Trustworthiness Viewpoint

The goal of the trustworthiness viewpoint is to address concerns related to security, data privacy and reliability of the NEPHELE platform. This viewpoint ensures that the NEPHELE platform is a secure ecosystem that can safely process and store sensitive information, preserve privacy, and adhere to industry regulations. This is achieved through implementing robust security measures, encryption protocols, and access controls to protect against potential threats and vulnerabilities, thus maintaining the trust among all the stakeholders. The NEPHELE Operator, CEP, NP, ASPs and Service Consumers are the stakeholders involved in this viewpoint.

4.2.5.1 Meta-Orchestration Platform Specifications

Table 13. Meta-orchestration platform specifications for trustworthiness viewpoint

Viewpoint Name	Trustworthiness viewpoint for the Meta-Orchestration Platform	
Overview	The Trustworthiness Viewpoint is dedicated to building and maintaining trust within the NEPHELE ecosystem. It addresses concerns related to security, data privacy, reliability, and ethical and legal compliance. This viewpoint ensures that the platform is trusted by both stakeholders and end-users.	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, IoT provider, NP
	Concerns	- Support local redundancy/high availability by endorsing a microservice and stateless approach for the architecture components. - Support fault tolerance mechanisms and assure reliability of the actions managed by the orchestration components. - Publish and exchange the required metrics, logs and KPIs in a secure and efficient manner, considering decentralized monitoring techniques. - Ensure that the workloads deployed throughout the continuum are always tracked and accessible to the service provider and consumer. - Offer a history of operations performed to each of the HDA components for its auditing. - Automatically scale resources to keep a controlled average CPU utilization measured at least every minute. - Enforce access control policies to prevent unauthorized users from allocating resources. - Employ anomaly detection to identify and mitigate any suspicious resource allocation activities. - Store access control policies, ensuring consistency in security configurations across deployments
	Model kinds/Legends	- Openness - Modularity - Interoperability
Related requirements (from D2.1 [11]): FR_SO_006, FR_SO_010, FR_SO_018, FR_SO_019, NFR_SO_003, NFR_SO_008, NFR_SO_009		

4.2.5.2 Virtual Object Stack (VOStack) Specifications

Table 14. VOStack specifications for the trustworthiness viewpoint

Viewpoint Name	Trustworthiness viewpoint for the VOStack	
Overview	The Trustworthiness Viewpoint is dedicated to building and maintaining trust within the NEPHELE ecosystem. It addresses concerns related to security, data privacy, reliability, and ethical and legal compliance. This viewpoint ensures that the VOStack is trusted by both stakeholders and end-users.	
Viewpoint specification	**Known typical stakeholders**	IoT provider, Application developer, CEP
	Concerns	- The VOStack shall provide secure connection mechanisms for the interaction between the VOs and the IoT devices. - The system should guarantee data security and privacy in transmission and storage. - The VOStack shall implement end-to-end encryption for data in transit and at rest. - It should also adhere to industry-specific data privacy regulations and undergo regular security audits and penetration testing. - The VOStack has to guarantee the reliable operation of (c)VOs and provide health checks. - Support secure communication protocols for the interaction between the VOs and the IoT devices (e.g., MQTT with TLS guarantees). - The VOStack shall be protected against emerging threats and vulnerabilities, through configuring effective security mechanisms.
	Model kinds/Legends	- Openness - Modularity - Interoperability
Related requirements (from D2.1 [11]): FR_VOS_009, NFR_VOS_01, NFR_VOS_02		

4.2.5.3 Dashboard and Development Environment Specifications

Table 15. Dashboard and Development environment specifications for the trustworthiness viewpoint

Viewpoint Name	Trustworthiness Viewpoint for the Dashboard and Development Environment
Overview	The Trustworthiness Viewpoint is dedicated to building and maintaining trust within the NEPHELE ecosystem. It addresses concerns related to security, data privacy, reliability, and ethical and legal compliance. This viewpoint ensures that the dashboard and development environment is trusted by both stakeholders and end-users.

Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, ASP, ASD, Service Consumers
	Concerns	- User credentials should be securely stored using specialized backend services which should be common for all the platforms. - All components should be accessible via a HTTPS connection. - HDAR should enable multi-tenancy and isolation of artifacts. - HDAR and the Development Environment should enable a way to provide verification of the artifacts searching for static errors such as syntax, semantics and structure issues with respect to the corresponding specification. - It should be possible to audit the usage of artifacts from the HDAR.
	Model kinds/Legends	- Openness - Modularity - Interoperability
Related requirements (from D2.1 [11]): FR_SO_015, FR_SO_017, NFR_VOS_02		

4.2.6. Construction Viewpoint

The construction viewpoint involves the effective co-design of the hardware and software components, aiming to provide resource management and network management, and ensure the security, reliability and scalability of the entire NEPHELE infrastructure. Stakeholders within this viewpoint, i.e., the NEPHELE Operator, CEP, NP, and ASPs, work diligently to construct and evolve the platform's technical foundation, enabling its growth and adaptability over time. Furthermore, aspects such as infrastructure maintenance and vitality across time should also be considered. In the dynamic landscape of IoT-to-Edge-to-Cloud computing, where the interplay of diverse stakeholders, intricate functionalities, and evolving technologies is constant, these processes are indispensable. The maintenance is crucial for identifying and addressing potential issues within the NEPHELE system, ranging from hardware and software components to configurations. Validation processes ensure that the NEPHELE system effectively meets its objectives and aligns with stakeholders' expectations, while Update processes are essential to keep the NEPHELE system aligned with evolving requirements, technological advancements, and security standards. Regular software updates, security patches, and advancements in functionalities are necessary to address vulnerabilities, enhance performance, and introduce new features. This systematic approach ensures that the NEPHELE system remains at the forefront of innovation, providing stakeholders with cutting-edge capabilities.

4.2.6.1 Meta-Orchestration Platform Specifications

Table 16. Meta-orchestration platform specifications for the construction viewpoint

Viewpoint Name	Construction viewpoint for the Meta-Orchestration Platform	
Overview	Considers the processes that have to be supported for the development, maintenance and update of the Meta-Orchestration Platform	
	Known typical stakeholders	NEPHELE Operator, CEP, NP, IoT Provider

Viewpoint specification	**Concerns**	- All components should be developed following a microservice architecture to ensure that an update in one component does not take down the whole platform. - Internal components of the NEPHELE Platform shall ensure high levels of service availability measured on a daily basis. - The Meta-Orchestrator's ability to handle different workloads and ensure optimal resource allocation should undergo performance testing simulations under various usage scenarios. - Generate test scenarios under different interference levels for discrete HDA deployments. - The Meta-Orchestrator platform should schedule automated algorithms during periods of low system load, considering historical usage patterns and performance metrics. - Ensure the continuous verification of the correct instantiation, deployment, and secure communication establishment and networking. - Validation of the Meta-Orchestration for diverse traffic patterns and changing workload dynamics, thus confirming the Network Resource Manager's adaptability to evolving conditions. - Ensure storage and retrieval of information related to instantiated VOs, including device descriptions, cluster affiliations, IPs, and communication protocols.
	Model kinds/Legends	- Software Development Repository - Continuous development, integration and testing approach - Open-source solutions
Related requirements (from D2.1 [11]): FR_SO_006, FR_SO_007, FR_SO_008, FR_SO_011, FR_SO_017, NFR_SO_003, NFR_SO_004, NFR_SO_005, NFR_SO_006		

4.2.6.2 Virtual Object Stack (VOStack) Specifications

Table 17. VOStack specifications for the construction viewpoint

Viewpoint Name	Construction viewpoint for the VOStack	
Overview	Considers the processes that have to be supported for the development, maintenance and update of the VOStack	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, IoT Provider
	Concerns	- All components should be developed following a microservice architecture to ensure that an update in one component does not take down the whole platform. - Internal components of the VOStack shall ensure high levels of service availability measured on a daily basis.

		- The TSN control plane in the VO Stack shall provide a generic NorthBound API using a well-defined JSON schema for application configuration and requirements processing. - The VO shall expose a particular type of SouthBound interface that can support SDN-based IoT devices - Clustering capabilities shall be supported at the cVO level for associating IoT nodes with VOs, configuring node-specific protocol settings, and implementing proactive routing. - The VOStack shall enable maintenance through regular updates to the application-oriented interfaces and address emerging communication challenges with Edge-Cloud entities. - In the physical layer, the VO should support expanding device support and device management refinement.
	Model kinds/Legends	- Software Development Repository - Continuous development, integration and testing approach - Open-source solutions
Related requirements (from D2.1 [11]): FR_VOS_010, NFR_VOS_03, NFR_VOS_04, NFR_VOS_05, NFR_VOS_06, NFR_VOS_07		

4.2.6.3 Dashboard and Development Environment Specifications

Table 18. Dashboard and Development environment specifications for the construction viewpoint

Viewpoint Name	Construction viewpoint for the Dashboard and Development environment	
Overview	The construction viewpoint is concerned with the technical aspects of building and maintaining the NEPHELE platform. It covers hardware and software development, network infrastructure, architectural components, and system maintenance. Stakeholders, in this view, focus on the construction and ongoing development of the platform.	
Viewpoint specification	**Known typical stakeholders**	NEPHELE Operator, CEP, NP, ASP, ASD
	Concerns	- All components should be developed following a microservice architecture to ensure that an update in one component does not take down the whole platform. - All components should try to remain as stateless as possible. - Components potentially dealing with the largest traffic should put in place load balancing and scaling mechanisms. - The HDAR should be accessible from all infrastructure providers. - How would the credentials of an individual user reach the infrastructure in order to securely pull the artifacts from the HDAR?

		- The Dashboard and HDAR should be able to be upgraded with new versions without losing the underlying state of the artifacts and users. - The Development Repository and Documentation should allow feature proposals and a way to provide patches from the community. - The Development Repository should enable a way to fork and customize the demo HDA. - The backend storage of the HDAR should be scalable to avoid its saturation. - The development repository should allow the effective maintenance, validation and update of the NEPHELE components.
	Model kinds/Legends	- Software Development Repository - HDA Registry - Continuous development, integration and testing approach - Open-source solutions
Related requirements (from D2.1 [11]): NFR_SO_005		

The effective maintenance, validation and update of all the NEPHELE components is dominant for the longevity of the NEPHELE system. Following a continuous development and integration approach, all the components should undergo continuous enhancement to adapt to evolving standards and requirements, ensuring the secure and efficient provision of the envisaged functionalities.

5. NEPHELE Architectural Description

The NEPHELE ecosystem, with its intricate co-existence of IoT, Edge and Cloud computing, poses several architectural challenges that demand careful consideration. Moreover, the diversity of stakeholders, each with unique interests, objectives and concerns, introduces the need for robust mechanisms to accommodate varying perspectives and requirements. This leads to notable challenges, which lie on the architectural design of the major components of NEPHELE's platform, regarding the functionality, scalability, interoperability, robustness and seamless collaboration among the Meta-Orchestration platform, the VOStack, and the Dashboard and Development environment. The dynamic nature of HDAs further complicates the architecture, requiring solutions that can efficiently handle fluctuating demands, network scalability, and time-sensitive functionalities. Addressing these challenges is crucial for constructing a resilient and future-proof NEPHELE architecture capable of meeting the evolving needs of stakeholders and ensuring the success of the entire ecosystem.

5.1 NEPHELE Architectural Approach

The layers and individual mechanisms within the framework, developed to tackle the challenges of orchestrating hyper-distributed applications in the computing continuum, are thoroughly examined. A system of systems approach is adopted where an upper-level entity has the responsibility for the overall lifecycle management of the deployment and runtime of a distributed application, while the control is distributed in various entities following a hierarchical approach, as shown in Figure 3, focusing on addressing the complexities of orchestration in the computing continuum. These challenges primarily stem from the intricate task of harmonizing optimization efforts across various levels of computing and network resources, as well as in data and functionalities required to manage end devices. The hierarchical framework is designed to navigate and streamline these challenges by providing a structured and adaptable approach to multi-level optimization in the context of the computing continuum. Also, all requirements described in Deliverable D2.1 [11] are tackled by the components of the layered NEPHELE architecture. As shown in Figure 3, the main components of the architecture are:

1. **Dashboard and Development Environment:** This component hosts the set of developed hyper-distributed applications, (c)VOs and the user-defined application graphs and intents. It also provides views to end users for navigation to the software artifacts and the deployments.
2. **Synergetic Meta-Orchestrator (SMO):** This is the main component of the NEPHELE architecture and is responsible for the overall orchestration of the hyper-distributed applications over the available infrastructure
3. **Network Resource Manager:** This component is responsible for network management functionalities across the compute continuum.
4. **Multi-Cluster Resource Manager:** This component is responsible for having an overall view of the available computing resources across the continuum. It manages multi-cluster resources based on proper resources' abstraction.
5. **Edge/Cloud Resource Managers:** These components are responsible for managing the deployment part and life-cycle of applications in the edge and cloud clusters respectively.
6. **VOStack:** The VO is considered to be the virtual counterpart/extension of an IoT device into the network infrastructure, also here we include the functionalities of the IoT software stack (VOstack) [13].

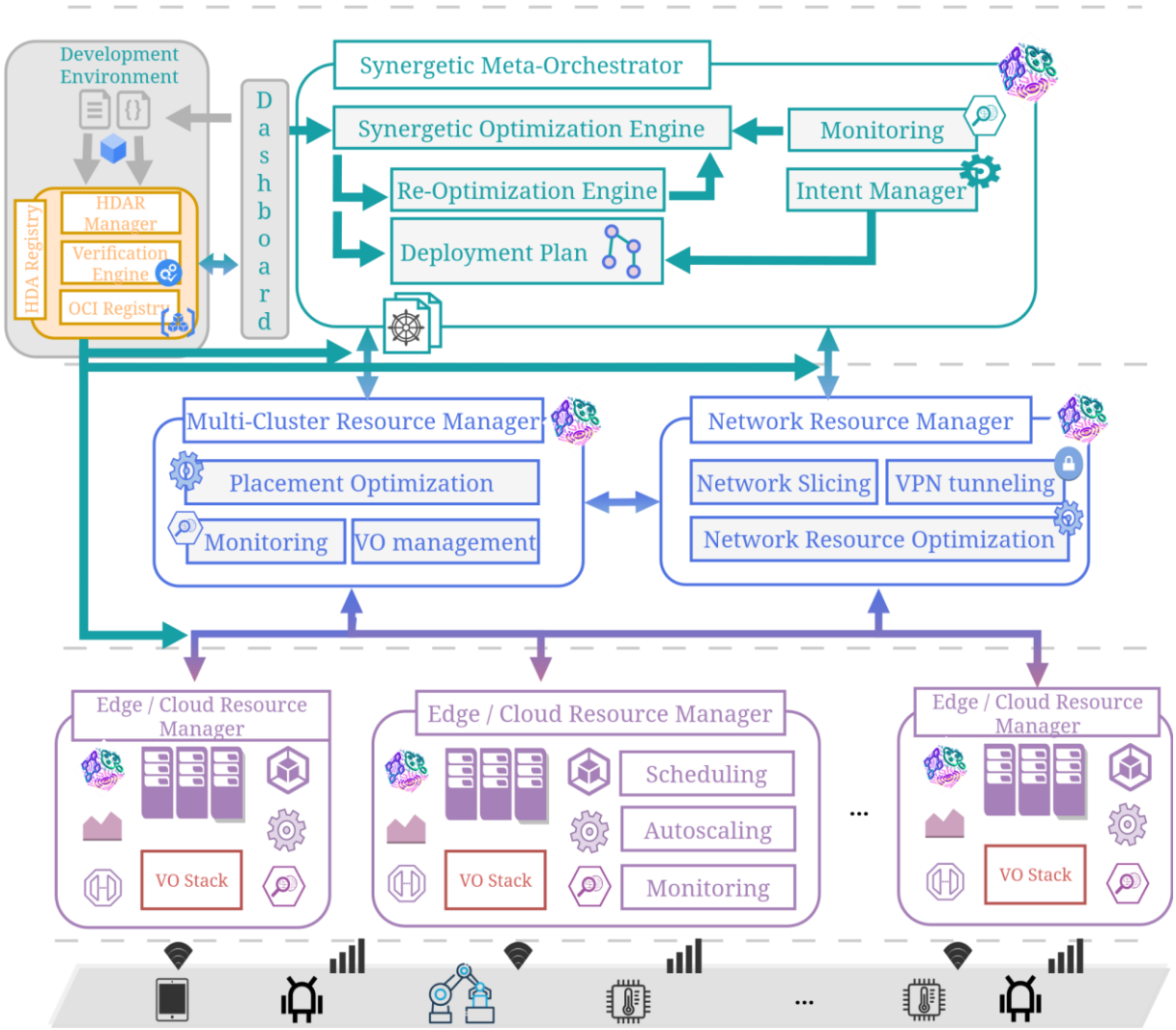

Figure 3. NEPHELE reference architecture

The application and infrastructure optimization are performed in a layered fashion, where specific events on each layer trigger the optimization mechanisms of others. In the following, functionalities and conceptual formulations of optimization problems addressed by each orchestration layer are described in detail. In what follows, we analyze the functionalities of each layer.

5.2.1. Dashboard and Development Environment

Hyper Distributed Applications are built on top of Cloud-Native and modern network management ecosystems and, as such, each one has their own set of technologies and development practices which enable the generation of the artifacts that are deployed in the infrastructure. The NEPHELE SMO framework has to be understood as an abstraction layer for the combined use of the aforementioned ecosystems and, for that, a series of Nephele-specific artifacts need to be generated and developed.

The Dashboard and Development Environment constitute the entry point to the NEPHELE SMO framework from a software development perspective and from a deployment management perspective respectively.

The **Development Environment** is internally divided in two systems. Both have been described in detail as part of D4.1 [12].

- First, a purely developer-oriented sandbox, which is based on the GitLab IDE and CI/CD framework, for the end-to-end development of the required artifacts. This covers the creation of Docker images, helm charts, NFV artifacts, (c)VOs, HDAGs (HDA Graphs).
- Second, the HDA Registry which provides, among other functionalities, the storage, distribution, and verification solution for all artifacts involved in the deployment of the HDA.

The **Dashboard** has been developed taking into account the needs of different stakeholders. On the one hand, it provides a central business and operations intelligence viewpoint where a user may obtain information about everything that they have access to in the SMO framework, including, monitoring of deployed HDAGs, catalog of available artifacts and resources in the infrastructure. On the other hand, it nicely supplements the development sandbox with a user-friendly UI to assist in the generation of the HDAG.

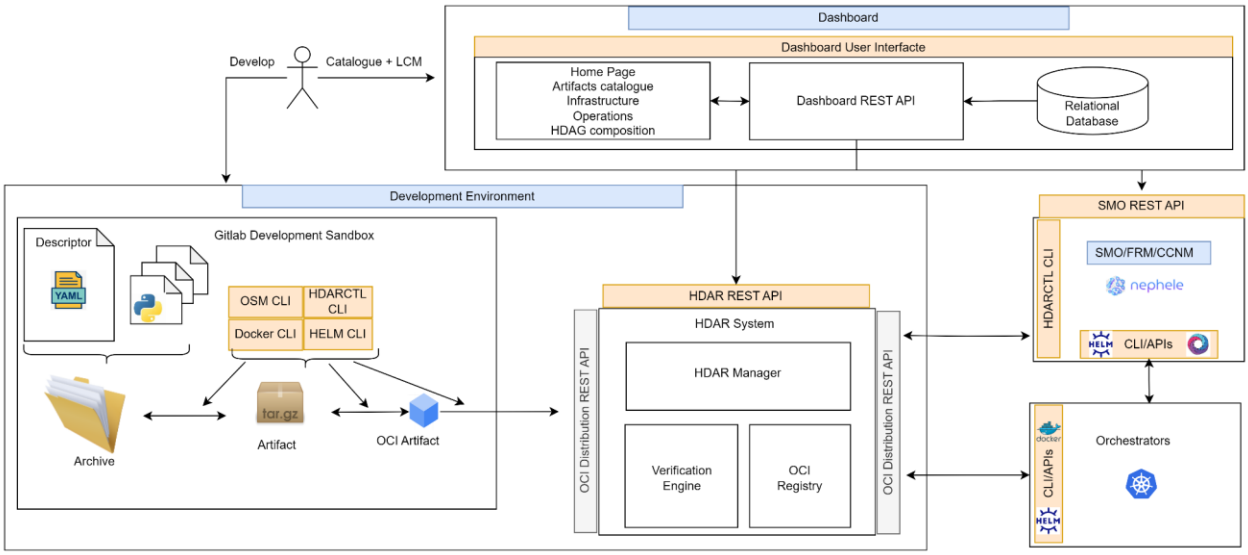

Figure 4. Dashboard and development environment of the Nephele SMO architecture as well as their main interfaces with other components

Figure 4 depicts all of the systems that are part of this component of the Nephele SMO architecture as well as their main interfaces with other components.

5.2.2. Synergetic Meta-Orchestrator

Orchestrating resources across the continuum demands optimization across various orchestration levels and throughout different phases of the application lifecycle. This optimization extends to ensuring the stable operation of distributed infrastructures. To address these demands effectively, a hierarchical structure of optimization mechanisms is necessary to enable the allocation of resources efficiently, the adaptation of orchestration strategies to changing conditions, and the enhancement of overall system performance and reliability [8], by perform the optimization in the appropriate when needed [9]. Besides, the developed mechanisms should work seamlessly with existing resource management standards and frameworks, such as Kubernetes[1], having the ability to collect monitoring data, but also to enforce optimization decisions. The SMO is the main entity that undertakes the responsibility for serving the deployment and lifecycle management of distributed applications over programmable resources in the computing continuum. In a nutshell, the SMO undertakes a deployment request through the dashboard and prepares a deployment plan by considering both the current status and available

[1] https://kubernetes.io/

resources of the underlying infrastructure as well as the user-defined requirements. Continuous monitoring and event-triggered optimization actions may be produced and forwarded to managers in the lower levels of the hierarchy in the system of systems approach. Specifically, the SMO interacts with the multi-cluster resource manager and the network resource manager. Details for each component follows.

Intent Manager/Translation: We consider an approach where the computational and network resources in the continuum are abstracted from the application developers and providers. Through resources abstraction, there is no need for the declaration of resources for the deployment of serverless applications, simplifying the deployment process, especially in cases of deployments across heterogeneous clusters. In this way, high level objectives, properties and constraints can be specified during development time. This set of information is considered as the deployment intent and can be associated with one or more Service Level Objectives (SLOs). Based on the described intent and the mapping with SLOs, orchestration mechanisms are developed that target to achieve the desired intent or state based on optimal deployment and runtime adjustments. To allow developers to express high-level requirements without specific technical details, we utilize intents [10]. The user-defined requirements are mapped to actual resources using an intent translation mechanism. There are various ways to handle this specific translation problem e.g., natural language processing or AI techniques. Moreover, the Intent Manager can be triggered again from the SMO to better optimize the translation of requirements to specific resources in case the user intents' fail to be satisfied.

Deployment Planner: At the SMO level, the application requirements submitted as intents, are processed to formulate a deployment plan, which, essentially, determines the requirements for an application graph deployment. Specifically, we consider that the intent requirements are mapped into specific categories for each application node, e.g., (i) CPU resources, (ii) memory resources, (iii) number of replicas, and (iv) network constraints. The mapping of requirements occurs through the interpretation of the intent at the SMO level, laying the foundation for defining and deploying an application graph. Therefore, upon the translation of the intent, the deployment plan for the application graph is realized. To construct a deployment plan, the SMO triggers the re-optimization engine in case the NEPHELE ecosystem is not able to host the realized deployment plan.

Re-optimization Engine (compute/network): This component interacts with the Network Resource Manager and the Multi-Cluster Resource Manager to query the availability in terms of network and computing resources and dictate the required actions to satisfy the application requirements, and facilitate the infrastructure management. Specifically, the Re-optimization Engine is responsible for triggering the Network Resource Manager to expand or scale down the NEPHELE infrastructure according to the current state and needs.

AI-assisted Synergetic Optimization Engine: This component is responsible for triggering of different levels of synergy between the layers of NEPHELE architecture based on events. Following the principle of system of systems, the SMO monitors (i) the entire NEPHELE infrastructure (compute/network) and (ii) specific performance metrics regarding the application graphs and triggers various events that enable the synergy between the components of the architecture. The Synergetic Optimization acts as a long-term decision loop to coordinate and optimize the use of resources, while also optimizing the performance of the application graphs coordinated by NEPHELE. An example overview of different levels of event-based synergy are presented in Section 5.3.

Centralized Monitoring Engine: The SMO monitors the NEPHELE infrastructure and during the lifetime of the application graph collects metrics from the deployed resources, the respective traffic metrics, and the performance metrics of the application nodes, which are stored in a centralized monitoring system (e.g. Prometheus[2]). Hence, various events and triggering mechanisms can be exposed

[2] https://prometheus.io/

using open-source tools (e.g., Grafana[3]) to enable the Synergetic Optimization workflows. This combination provides a rich, interactive experience for exploring data through dashboards, making it simpler to identify trends and troubleshoot issues. In this sense, the Centralized Monitoring Engine also facilitates the Dashboard, as discussed previously, to allow NEPHELE users monitor and validate the performance of their application graphs.

5.2.3. Network Resource Manager

The Network Resource Manager is responsible for managing network resources and functionalities across the edge-cloud compute continuum and ensuring network performance guarantees for distributed application deployments. This role encompasses several critical points that have to be addressed by the individual mechanisms of this component. Network Resource Manager undertakes the identification of the computing and network resources and the setup of the related cluster topology, by instantiating network slices according to the QoS requirements of the considered application, followed by the management of their lifecycle, while encompassing processes for the ongoing optimization of network resource utilization. Moreover, resources across the continuum are geographically distributed among multiple clusters. Given the characteristics of the computing continuum ecosystem and the complexity and dynamic nature of distributed application deployments, besides the establishment of robust and secure communication channels between different clusters, Network Resource Manager should, also, focus on continuous adaptation of the topology configurations to accommodate changing traffic patterns and workload dynamics.

These essential requirements determine the key functionalities of the component, which are undertaken by the corresponding core mechanisms that is composed of:

Network Slice Lifecycle Management: This encompasses the creation, deployment, monitoring, and termination of network slices. The lifecycle management ensures that network resources are provisioned and configured according to application needs, optimizing performance and resource utilization.

VPN tunneling: VPN tunneling establishes secure and encrypted communication channels over public networks, enabling private connectivity between distributed clusters. Through VPN tunneling, the Network Resource Manager ensures the integrity, and authentication of data transmitted between distributed clusters, safeguarding sensitive information against unauthorized access or interception. This layer forms a secure communication channel in distributed environments, bolstering the resilience and trustworthiness of interconnected clusters.

Network Resources Optimization Engine: This component continuously monitors, analyzes, and optimizes network resource utilization. By leveraging real-time monitoring data regarding application performance and infrastructure utilization, and Network Function Virtualization capabilities, this optimization engine orchestrates the efficient allocation and management of network resources by dynamically adjusting network configurations, routing policies, and bandwidth allocation to enhance performance, reliability, and efficiency.

Infrastructure Scaling: In scenarios where infrastructure scaling is required, the Network Resource Manager seamlessly integrates with Virtualized Infrastructure Manager and Resource Orchestration frameworks like OpenStack[4] and Kubernetes, to facilitate seamless expansion or contraction of the utilized resources. Leveraging these platforms, it orchestrates the provisioning and scaling of infrastructure and connectivity components, ensuring that the network infrastructure remains agile and responsive to changing demands. Whether scaling-out to accommodate sudden surges in traffic or scaling-in to conserve resources during periods of low demand, the Network Resource Manager

[3] https://grafana.com/

[4] https://www.openstack.org/

dynamically adjusts network configurations to maintain optimal performance and resiliency of the distributed cluster topology.

5.2.4. Multi Cluster Resource Manager

The Multi-Cluster Manager oversees the management of computational resources across multiple Edge/Cloud clusters. This component operates under the assumption that various clusters are integrated into a unified continuum. These clusters may encompass a spectrum from IoT devices to edge computing to the cloud infrastructure. Therefore, the Multi-Cluster Resource Manager/Orchestrator manages multiple (Kubernetes-based) clusters across diverse environments and providers, offering a centralized and unified interface for administration and operation. This centralized control simplifies the management process, reduces the complexity of handling disparate systems, and minimizes the potential for errors enhancing the overall efficiency. The Multi-Cluster Manager facilitates actions related to placement, security, migration, elasticity, and Virtual Object discovery across the interconnected clusters. Furthermore, it incorporates a federated monitoring engine that combines data from diverse monitoring engines operating at the cluster level.

Placement Optimization Engine: With the deployment plan from the SMO in place, the distributed placement of nodes of an application graph becomes a specific optimization challenge. The Placement Optimization component acts as an Integrated mechanism of the Multi-Cluster Manager to address this challenge. Specifically, ensuring the connectivity among the available clusters, the placement optimization takes place in the Multi-Cluster Resource Manager, to guarantee the application requirements with efficient resource utilization among the involved clusters. As a result, with respect to the intent acting as a constraint this optimization mechanism decides the appropriate Cluster to be deployed depending on the availability, the requested resources (e.g., CPU, memory, GPU) and the network constraints (e.g., bandwidth, collocation constraints), for each application node of the application graph. Acting as a centralized entity for all available clusters the Multi-Cluster Manager communicates via specific APIs with the Cluster Managers to propagate this decision and instantiate the application graph. The multi-cluster placement optimization could be triggered, also, in cases where re-optimization of the application graph placement is needed.

Centralized Security Mechanism: This component is known as the 'Issuer' in the security architecture and acts as the trusted entity for issuing the Verifiable Credentials (VCs) using the OpenID Connect for Credential Issuance (OIDC4CI), an extension of the OpenID Connect (OIDC[5]) protocol in which a Holder (e.g., a VO or a cVO) interacts with the Issuer to obtain the VC, which is signed with the Issuer's private key. This private key is associated with a public key, both belonging to a Decentralized Identifier (DID) that uniquely identifies the Issuer. The Issuer's DID must be known in every component of the architecture that participates in security mechanisms as it will verify that a particular VC has been issued by the Issuer of the architecture and is therefore a VC that can be trusted. This centralized approach enhances security by providing a single point for issuing credentials, providing integrity, availability, confidentiality, and trust.

VO Discovery Server: This component stores all necessary information about the instantiated VOs and the respective devices that the VOs represent. Specifically, the description of the device, the cluster where the VO is deployed, the IPs and all necessary information and metadata (e.g., communication protocols). As VOs can be part of more than one application graph, this component is responsible for communicating with the placement optimization engine for the above-mentioned information and also facilitates the operations of Centralized Security Mechanism for the secure communication between applications and VOs.

[5] https://openid.net/

Multi-Cluster Monitoring: This component stores metrics, traces, and logs that constitute a set of observability signals from all the clusters in the NEPHELE ecosystem. This provides a fully observable view of the deployed applications and their environment and offers a framework enabling the other functionalities in the Multi Cluster Resource Manager level.

5.2.5. Edge/Cloud Resource Manager

At the cluster level, an individual Cluster Manager (CM) is activated for each edge or cloud cluster, overseeing the deployment of a segment of the application graph within its designated infrastructure. This involves executing local orchestration actions and gathering monitoring information. A Cluster Manager can be tasked with managing resources in diverse environments, such as a cloud computing cluster, an edge computing cluster, or overseeing resources within a powerful IoT device or a cluster of IoT devices. Each application node will be deployed as a pod and the respective service in the dedicated Kubernetes-based Cluster. Also, specific optimization techniques are employed to ensure the fulfillment of application graph requirements at the cluster level. These techniques focus on achieving efficient resource scheduling and implementing autoscaling mechanisms for the corresponding application nodes, to meet performance criteria while minimizing costs, both during the application deployment and on real-time workload fluctuations. The components of the Cluster Resource Manager are analyzed in what follows.

Cluster Scheduler: This component is responsible for implementing the actual scheduling of application nodes to the underlying infrastructure of each Kubernetes cluster. Therefore, in order to orchestrate the scheduling of the replicas of the application nodes to the edge and cloud clusters, we will formulate a multi-objective optimization problem that will compute a scheduling strategy in order to satisfy the systems' intents. Indicatively, we will try to minimize (a) the link utilization for the intra-cluster communication (b) the transformation cost for the application nodes in case of migration between different Kubernetes worker nodes, which indirectly minimizes the delay by reducing the number of cold starts of the applications nodes, and (c) the total power utilization of the underlying infrastructure.

Cluster Autoscaler: This component is responsible for deploying alongside the application the respective scaling mechanism that suits to each application depending on the intent translation mechanism. Based on different application requirements the Cluster Autoscaler instantiates the respective scaler (e.g., Horizontal Pod Autoscaler) that scales the deployed services according to the selected metrics based also on real-time monitoring and prediction of the incoming workload. Specifically, various mechanisms can be employed. NEPHELE aims to incorporate Reinforcement Learning (RL)-driven auto scaling techniques to compute the number of replicas for each application. This way, we guarantee that the average delay of the estimated requests is under a predefined threshold, while the number of deployed replicas is minimized avoiding the overprovisioning of resources.
Cluster Connectivity Manager: This component flattens the networks between the connected clusters, and enables IP reachability between Kubernetes pods and services. This way the application nodes communicate securely between them exposing only the necessary services for the inter-cluster communication utilizing e.g., VXLAN tunnels or VPNs. Hence, all the necessary connections between the application nodes can be done using DNS resolution.

Cluster Monitoring: Prometheus is a popular choice for cluster monitoring. Provides comprehensive data collection (metrics) and analysis capabilities. It collects default metrics (e.g. CPU and memory usage) for each node and each component in the cluster, but also enables the collection of custom metrics that are critical for understanding the unique aspects of an application's performance.

5.2.6. Virtual Object Stack

The Virtual Object (VO), is the virtual counterpart/extension of IoT devices deployed on the premises of Edge/Cloud Clusters. The VO is a lightweight software stack[6], based on two different specifications, W3C Web of Things[7] and Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LwM2M)[8], that stores the necessary information produced by IoT devices and therefore acts as a broker between the devices and an application graph. The purpose of the VO is to solve three main challenges, namely; (i) Semantic abstraction: It provides a standard representation of physical device for easier management, monitoring, and discovery of device resources, (ii) interoperability by implementing various communication protocols (HTTP, CoAP, MQTT), (iii) secure communication by deploying a set of authentication mechanisms (TLS, basic/bearer authentication) and (iv) enriching the capabilities of resource-constrained devices by deploying a set of advanced functionalities (e.g compression, data series analysis etc.) and (v) data modeling enabling the use of different data modeling for different purposes leveraging physical device from complex data model transmission. Moreover, the VO can also be used as a Digital Twin and can be used to simulate the physical device behavior in a virtual environment before deployment. Using the VOStack an application can trigger actions or subscribe to events produced from IoT devices. As a result, the VO is responsible for deploying event processing mechanisms. Also, the IoT device can offload parts of its computing burden to the VO, deploying a set of advanced functionalities.

Moreover, a Composite Virtual Object (cVO) is a software entity that is able to manage the information coming from one or multiple VOs and provide advanced functionalities.

Therefore, the proposed stack addresses IoT interoperability and openness aspects at two different levels, namely (i) the IoT device level, based on the provision of virtual counterparts of IoT devices, and (ii) the level of integration of IoT functions with edge and cloud computing applications. Nonetheless, a (c)VO can also tackle multi-tenancy problems meaning that a VO can be part of more than one application graph.

In order to support the features described above, the NEPHELE project aims to develop a complete software stack, VOStack, which is divided in three levels, as shown in Figure 5.

- **Edge/Cloud convergence:** This layer of the VOStack is application-oriented and it oversees the management of all interfaces that interact in communications with other edge-cloud entities present in the NEPHELE platform like applications and orchestrators that intend to interact with the IoT physical domain and virtualized services.
- **Backend logics** (for IoT functionalities): This layer is internal to the VO, and it does not expose interface to prosumers. The layer deals with implementing the enhancement of VO functionalities with respect to the physical device
- **Physical convergence:** This layer is responsible for solving challenges to allow the interconnection, interoperability, and device management of as many as possible, if not all, constrained and non-constrained IoT devices.

[6] https://netmode.gitlab.io/vo-wot/

[7] https://www.w3.org/WoT/

[8] https://technical.openmobilealliance.org/index.html

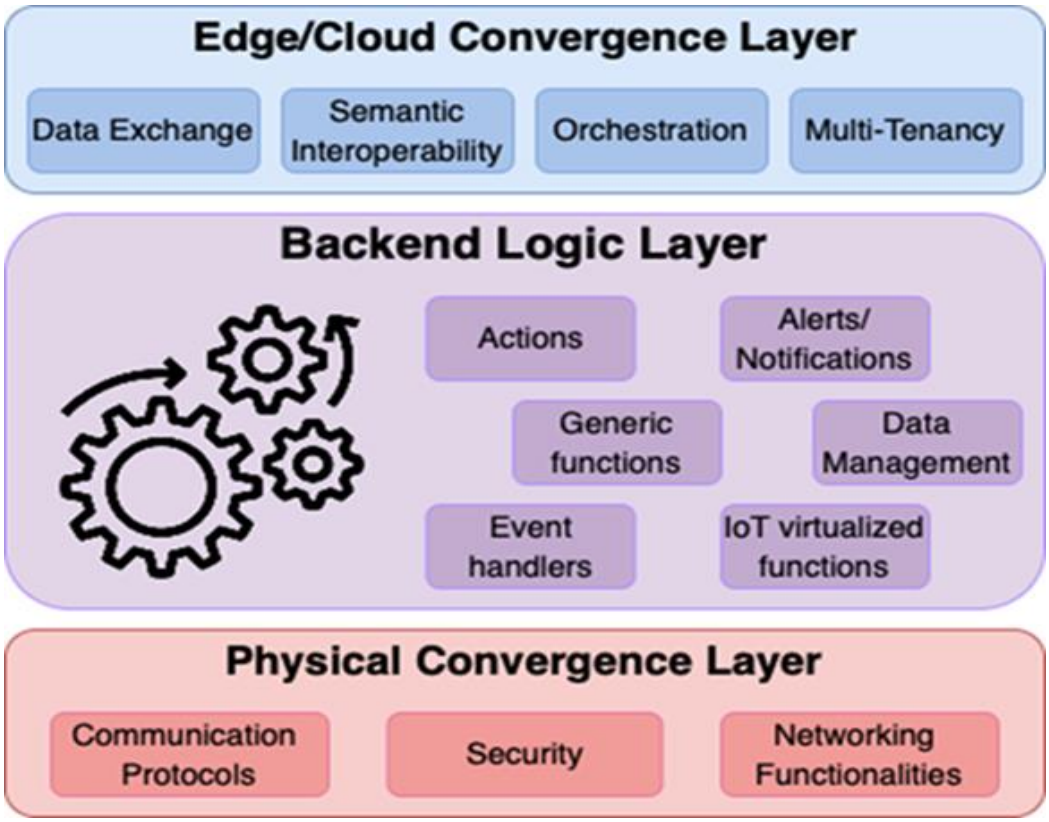

Figure 5. Virtual Object Stack (VOStack) Layers

Details for each layer of the (c)VO can be found in D.3.1 [13] along with the respective functionalities.

5.3 Overview of the per Layer Decision Making of the NEPHELE Platform

Here we define some scenarios that showcase the synergy and interaction between the different layers and components of the NEPHELE architecture. We aim to extend and enrich these scenarios and implement the respective triggering methodology at the SMO level as described previously. The specific workflows for the synergy between different components are illustrated in Figure 6.

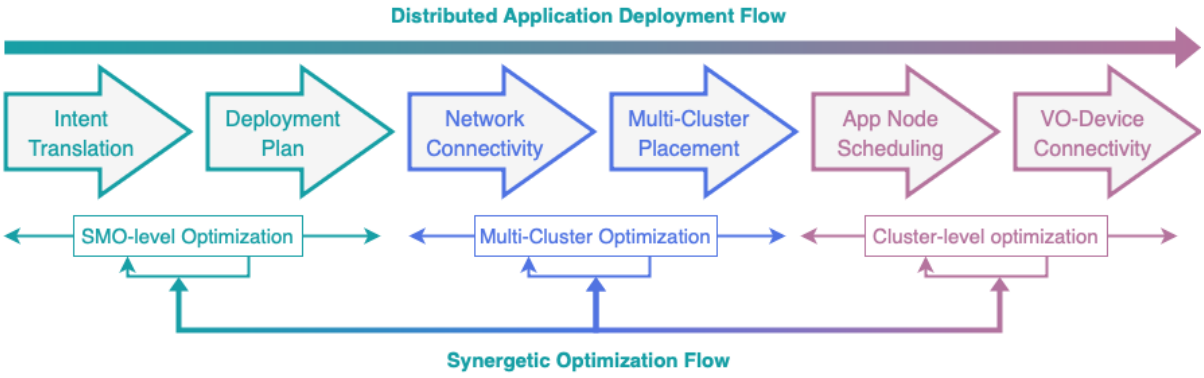

Figure 6. Example of Synergetic Optimization Flow between different NEPHELE components

We consider the following events that trigger synergetic orchestration between the components of NEPHELE.

1. **Deployment of a new application graph:** This is the most basic event (illustrated with A in Figure 7) in the NEPHELE ecosystem. Following the deployment of a new application graph through the dashboard and the development environment, the SMO is triggered to instantiate the services depending also on the users' intent. Specifically, the AI-assisted Synergetic Optimization Engine triggers the Intent Manager to translate the intent into specific requirements. Depending on the requirements the Network Resource Manager is triggered to facilitate the scaling in/out of the infrastructure. Similarly, the Multi-Cluster Resource Manager may carry out a re-optimization of the placement of application graphs depending on the scaling of the NEPHELE infrastructure. Then, the deployment plan is triggered for the new application graph to facilitate the correct workflow of the application graph. Also, the Re-optimization Engine adapts the topology and triggers again the Placement Optimization Engine to assist in the scheduling of the new application graph in the operating edge/cloud clusters.
2. **Performance of deployed application graph:** Similarly to the previous event, when the Centralized Monitoring Engine triggers a performance alert for a specific application graph (illustrated as B in Figure 7), then two possible options are considered. First, the recalibration of the intent to meet the performance requirements (event B1) along with the necessary adjustments of the topology and replacement of the whole application graph if needed, Second, triggering a deployment re-plan (event B2) that basically means that the deployed resources are adjusted (e.g., an application node that had 2vCPUs, now it is assigned with GPU acceleration) depending on the application performance. Accordingly, the Re-optimization Engine adapts the topology (if necessary) and triggers also the placement of the refined application graph. Moreover, similar events can be triggered by Edge/Cloud Resource Manager levels (illustrated as F, G in Figure 7) for alerting about bad performance of the scaling or the scheduling of a specific application. This is tackled at the cluster level by adapting the mechanisms to compensate for the performance degradation.
3. **Performance of NEPHELE infrastructure:** Another event is triggered when the Centralized Monitoring Engine observes discrepancies in the performance of the infrastructure (illustrated as C in Figure 7), e.g., high/low CPU usage of clusters, high/low availability of resources. Then again two different actions are considered: the Infrastructure Scaling as described previously (illustrated as C1) or the Load Balancing of applications to achieve balanced utilization of resources triggering the placement mechanism (illustrated as C2). Moreover, we consider that a similar event, i.e., a cluster scheduling performance alert (illustrated as E in Figure 7), can be triggered by the Multi-Cluster Monitoring engine that can trigger the refinement of the placement of the application graphs to meet the performance demands.
4. **VO migration:** Finally, the last event considered in the NEPHELE ecosystem is a VO migration event (illustrated as D in Figure 7), triggered by end users through the dashboard. As we considered that a VO must be "close" in terms of network proximity with the respective device, then the Placement Optimization Engine is responsible to adapt the cluster where the respective VO is deployed.

The above defined events will be used to develop a strategy that will be dictated by the AI-assisted Synergetic Optimization Engine. As these events may be coupled or the actions triggered by each respective event may affect others, we aim to develop a mechanism at the SMO level that will device and have the overall assessment of what action is needed depending on various parameters such as, historical patterns (e.g., failing of consecutive scaling decisions at the cluster level), network metrics, application workload, interference between deployed applications, refinement of intents etc. The correlation of metrics and KPIs and the synergy between the different layers of the NEPHELE infrastructure can be done by trustworthy AI/ML techniques. To this extent, we will also consider the following high-level optimization goals, namely; (a) power utilization of clusters and overall cost reduction of infrastructure (b) performance guarantees of application graphs (c) high accuracy of the

intent translation mechanism, (d) the load balancing of usage between the edge/cloud clusters to achieve overall resilience, (e) real-time elasticity (f) live migration with minimal downtime (g) trust and security of data exchange through the VOs and finally (h) network isolation and slicing to achieve secure communication.

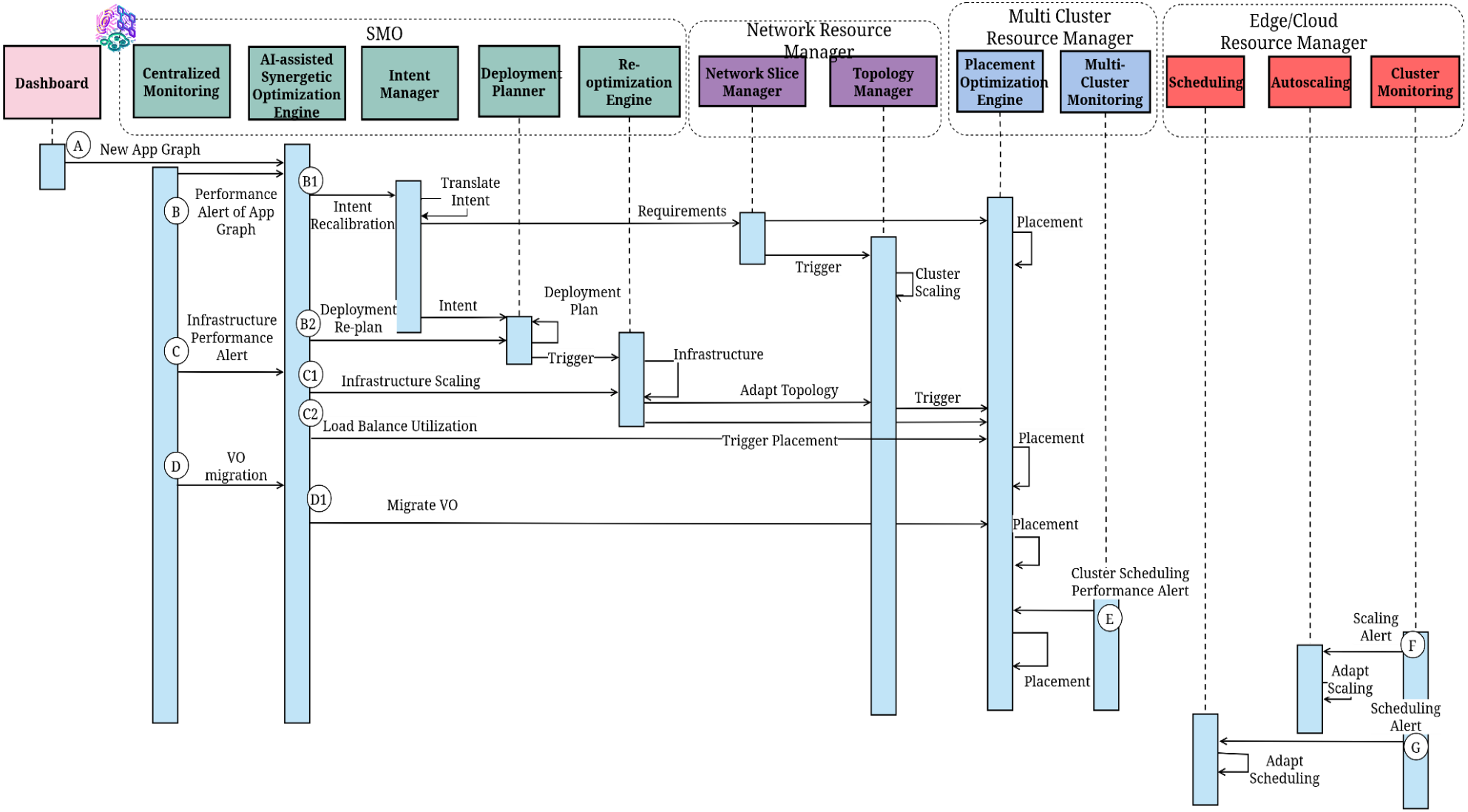

Figure 7. Event-based Synergetic Optimization of NEPHELE

5.4 Interaction Workflow of Hyper Distributed Applications

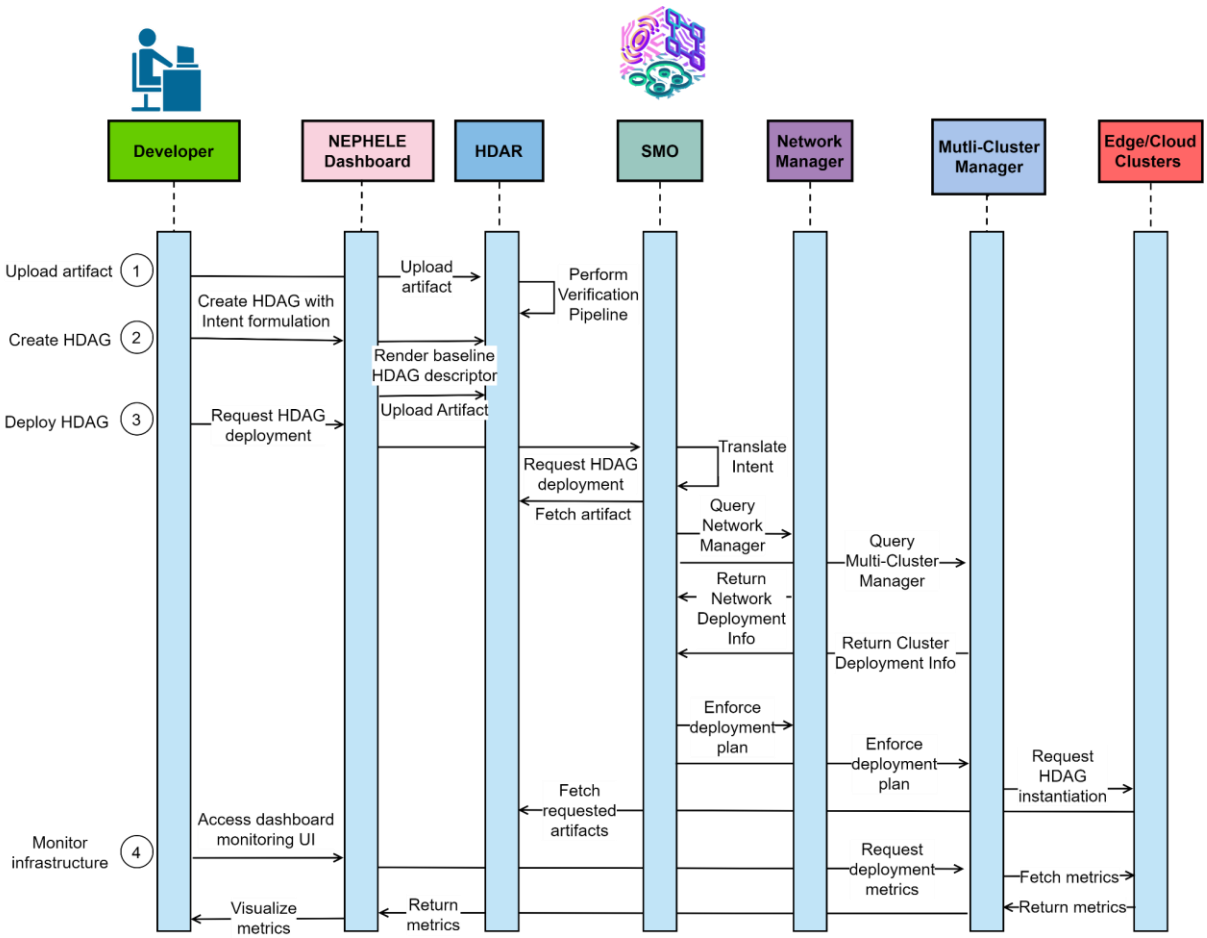

Figure 8. Interaction workflow among the NEPHELE components

To better assist the reader with the deployment of an application graph in this section we highlight the interaction of all components of the NEPHELE architecture. Firstly, the application is developed and packaged in a software artifact. The artifact is then uploaded to the Hyper Distributed Application Registry (HDAR) so that it can be referenced in Application Graphs. Afterwards, the Hyper Distributed Application Graph is constructed specifying the details of the various nodes that compose the Graph along with an intent formulation describing the desired application performance and its constraints. Once the Graph has been submitted to the Dashboard it is ready for deployment. After deployment of the graph is requested, the dashboard forwards the request to the SMO. In turn the SMO needs to communicate with the Network Manager and the Multi-Cluster Manager to calculate specific deployment information needed for the instantiation of the graph while the SMO itself translates the intent formulation into specific deployment instructions. Once the deployment plan is constructed, it is propagated to both the Network Manager and the Multi-Cluster Manager so that they can deploy all the necessary resources. More specifically the deployed applications fetch from the HDAR the artifacts that were declared in the Application Graph. Lastly, the dashboard can be used to supervise the deployment of the application graph and measure performance metrics and data. The full workflow can be seen in Figure 8.

6. Conclusions

In this document, we have presented an overview of the NEPHELE architectural description. Our description relies on the ISO/IEC/IEEE 42010 standard, which addresses the creation, analysis and sustainment of architectures of systems through the use of architecture descriptions. Our architecture is described based on three major components of the NEPHELE system: i) the Meta-Orchestration platform, ii) the VOStack and iii) the Dashboard and Development Environment.

Based on the ISO/IEC/IEEE 42010 standard, we first present an overview of its major components and the interoperability among them. We utilize the standard for effectively describing the fundamental principles of the standard, i.e. stakeholders, concerns, views, and viewpoints. These form a structured framework within ISO/IEC/IEEE 42010, enabling effective communication and documentation of architectural decisions in a manner that is both comprehensive and accessible to diverse stakeholders. We present six discrete stakeholders for the NEPHELE system, i.e. NEPHELE operator, Cloud-Edge Provider, IoT Provider, Network Provider, Application Service Developer/Provider and Service Consumer and we present a detailed analysis of their individual objectives and concerns, thus forming their perspective. Based on the latter, the viewpoints provide a comprehensive understanding of the architecture, allowing for effective communication, analysis, and decision-making throughout the system's lifecycle. We present and analyze the NEPHELE major components from foundational, business, usage, functional, trustworthiness and construction viewpoints.

Following the outcomes of the ISO/IEC/IEEE 42010 standard, the architectural description of the NEPHELE ecosystem is presented. The major objective is to derive the architectural design of the major components of NEPHELE's platform, regarding the functionality, scalability, interoperability, robustness and seamless collaboration among the Meta-Orchestration platform, the VOStack, the Dashboard and Development environment.

This deliverable will serve as a reference document to the other WPs and deliverables of the project since it presents all the main information about the NEPHELE's architecture and design. This document will be the generic architectural guideline for the development of the VOStack in WP3, the Meta-Orchestration Platform and the Development environment in WP4, and the integrated NEPHELE platform in WP5.

References

[1] Tamiru, Mulugeta Ayalew, et al. "mck8s: An orchestration platform for geo-distributed multi-cluster environments." *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021.

[2] Ullah, Amjad, et al. "Orchestration in the Cloud-to-Things compute continuum: taxonomy, survey and future directions." *Journal of Cloud Computing* 12.1 (2023): 135.

[3] Costa, B., Bachiega Jr, J., de Carvalho, L. R., & Araujo, A. P. (2022). Orchestration in fog computing: A comprehensive survey. *ACM Computing Surveys (CSUR)*, *55*(2), 1-34.

[4] Sarrigiannis, I., Antonopoulos, A., Ramantas, K., Efthymiopoulou, M., Contreras, L. M., & Verikoukis, C. (2022). Cost-Aware Placement and Enhanced Lifecycle Management of Service Function Chains in a Multidomain 5G Architecture. *IEEE Transactions on Network and Service Management*, *19*(4), 5006-5020.

[5] Toka, L., Dobreff, G., Fodor, B., & Sonkoly, B. (2021). Machine learning-based scaling management for kubernetes edge clusters. *IEEE Transactions on Network and Service Management*, *18*(1), 958-972.

[6] Filinis, N., Tzanettis, I., Spatharakis, D., Fotopoulou, E., Dimolitsas, I., Zafeiropoulos, A., Vassilakis, C. & Papavassiliou, S. (2024). Intent-driven orchestration of serverless applications in the computing continuum. *Future Generation Computer Systems*, *154*, 72-86.

[7] Spatharakis, D., Dimolitsas, I., Genovese, G., Tzanettis, I., Filinis, N., Fotopoulou, E., ... & Papavassiliou, S. (2023, June). A Lightweight Software Stack for IoT Interoperability within the Computing Continuum. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)* (pp. 715-722). IEEE.

[8] Kokkonen, Henna, et al. "Autonomy and intelligence in the computing continuum: Challenges, enablers, and future directions for orchestration." *arXiv preprint arXiv:2205.01423* (2022).

[9] Kimovski, Dragi, et al. "Cloud, fog, or edge: Where to compute?." *IEEE Internet Computing* 25.4 (2021): 30-36.

[10] Leivadeas, Aris, and Matthias Falkner. "A survey on intent based networking." *IEEE Communications Surveys & Tutorials* (2022)

[11] NEPHELE Deliverable D2.1: “Requirements, Use Cases Description and Conceptualization of the NEPHELE Reference Architecture”

[12] NEPHELE Deliverable D4.1: “Initial Release of Hyper-distributed Applications Synergetic Meta-Orchestration Framework, Development Environment and Repository”

[13] NEPHELE Deliverable D3.1: “Initial Release of VOStack Layers and Intelligence Mechanisms on IoT Devices”