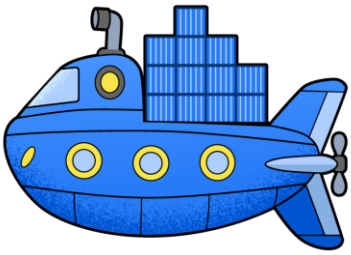


NEMO

Next Generation Meta Operating System

D1.3 NEMO meta-architecture, components and benchmarking. Final version

Document Identification			
Status	Final	Due Date	31/08/2024
Version	1.1	Submission Date	03/09/2024

Related WP	WP1	Document Reference	D1.3
Related Deliverable(s)	D1.1, D1.2	Dissemination Level (*)	PU
Lead Participant	SPH	Lead Author	Nikos Drosos (SPACE)
Contributors	SIM, INTRA, MAG, SPACE, TID, WIND3, CONTI, ESOFT, ASM, FHW, SYN, COMS	Reviewers	Sonja Wächter, CONTI Ruben Ramiro, ATOS

Keywords:
MetaOS, meta-architecture, architecture, verification, validation, multi-cluster orchestration, workload, resource, cloud computing, edge computing, artificial intelligence, next-generation internet, Internet of Things, IoT, security, DevOps, 5G, TSN, multi-domain

Document Information

List of Contributors	
Name	Partner
Andreea Paunescu	SIM
Mircea Vasile	SIM
Dimitrios Skias	INTRA
Panos Karkazis	MAG
Astik Sammal	MAG
Albero Casanova	MAG
Nikos Drosos	SPACE
Emmanouil Bakiris	SPACE
Alejandro Muñiz Da Costa	TID
Luis Miguel Contreras	TID
Fabrizio Brasca	WIND3
Gianluca Rizzi	WIND3
Sonja Wächter	CONTI
Antonis Gonos	ESOFT
Prashanth Pedholla	ASM
Dimitrios Christopoulos	FHW
Theodore Zahariadis	SYN
Terpsi Velivassaki	SYN
Miha Smolnikar	COMS

Document History			
Version	Date	Change editors	Changes
0.1	11/06/2024	N. Drosos (SPH)	ToC
0.2	21/06/2024	A. Paunescu, M. Vasile (SIM), N. Drosos, Emm. Bakiris (SPH), T. Velivassaki, Th. Zahariadis (SYN)	Updates on NEMO Use Cases and final architecture
0.3	11/07/2024	P. Karkazis (MAG), D. Christopoulos (FHW)	Refined structure for section 4, Add first version of all text for Smart City & XR trial
0.4	19/7/2024	P. Karkazis, A. Casanova, A. Sammal (MAG), S. Wächter (CONTI)	Contribution in section 4, Update on Smart Industry use cases
0.5	01/08/2024	T. Velivassaki, (SYN), A. Muñiz Da Costa, L. M. Contreras (TID), Emm. Bakiris	Updates on final architecture, updates on Smart Farming & Smart Energy use cases

		(SPH), A. Gonos (ESOFT), P. Pedholla (ASM), M. Smolnikar (COMS)	
0.6	06/08/2024	T. Velivassaki, (SYN), N. Drosos (SPH)	Updates on EUCEI alignment
0.6.1	06/08/2024	S. Wächter (CONTI)	Peer review
0.6.2	14/08/2024	R. Ramiro (ATOS)	Peer review
0.7	03/09/2024	N. Drosos (SPH)	Consolidated version
1.0	03/09/2024	N. Drosos (SPH)	FINAL VERSION TO BE SUBMITTED
1.1	03/09/2024	R. Valle (ATOS)	Quality review and submission to EC

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	N. Drosos (SPH)	03/09/2024
Quality manager	R. Valle Soriano (ATOS)	03/09/2024
Project Coordinator	E. Pere Pages Montanera (ATOS)	03/09/2024
Technical Manager	T. Velivassaki (SYN)	03/09/2024

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AAA	Authentication, Authorization, and Accounting
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
CFDRL	Cybersecure Federated Deep Reinforcement Learning
CMDT	Cybersecure Microservices' Digital Twin
Cobot	Collaborative Robot
CPU	Central Processing Unit
DLT	Distributed Ledger Technology
DoA	Description of Action
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
EUCEI	European Cloud Edge IoT (community)
EV	Electric Vehicle
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HPC	High Performance Computer
HTTP	Hypertext Transfer Protocol
HW	Hardware
IaC	Infrastructure as Code
IAM	Identity and Access Management
IAS	Intent-based API/SDK
IDSA	International Data Spaces Association
IIoT	Industrial Internet of Things
IMC	Intent-based Migration Controller
IoT	Internet of Things
K8s	Kubernetes
KPI	Key Performance Indicator
LCM	Life-Cycle Manager
LDAP	Lightweight Directory Access Protocol
LSP	Large Scale Pilot
MAC	Mandatory Access Control
MAF	Meta-Architecture Framework
MANO	Management and Orchestration
MEC	Multi-access Edge Computing
metaOS	Meta-Operating System
ML	Machine Learning
mNCC	Meta Network Cluster Controller
MOCA	Monetization and Consensus-based Accountability

Abbreviation / acronym	Description
MQTT	Message Queuing Telemetry Transport
NFV	Network Function Virtualization
NGIoT	Next-Generation IoT
NGSI	Next Generation Service Interface
OS	Operating System
PMU	Phasor Measurement Unit
PPEF	PRESS & Policy Enforcement Framework
PRESS	Privacy, data pRotection, Ethics, Security & Societal
RA	Reference Architecture
RAM	Random Access Memory
RAN	Radio Access Network
RBAC	Role-Based Access Control
RL	Reinforcement Learning
SBOM	Software Bill of Materials
SDK	Software Development Kit
SDN	Software Defined Networking
SEE	Secure Execution Environment
SEM	Smart Energy & smart Mobility
SLA	Service Level Agreement
SLO	Service Level Objective
SSO	Single Sign-On
TSN	Time Sensitive Networks
UC	Use Case
V&V	Validation & Verification
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VR	Virtual Reality
WP	Work Package
XR	Extended Reality
YAML	Yet Another Markup Language

Executive Summary

NEMO aims to develop the meta–Operating System (metaOS), which will enable multi-cluster and multi-network orchestration of containerized workloads across the Internet of Things (IoT), edge and cloud continuum. As a (meta-)OS, NEMO will be user-centric, facilitating users to develop and deploy on top of NEMO. Moreover, NEMO will enable cloud and infrastructure providers to integrate their computing and networking resources into NEMO's infrastructure.

The present document provides the updated specifications of the NEMO architecture for building the envisioned metaOS, based on the initial version presented in the deliverable document D1.2 [1]. The main outcomes reported in this deliverable include:

- Refinement of functional and non-functional requirements of the NEMO metaOS, as a result of revisiting use case definitions and specifications, following the outcomes of pilot preparations and initial results.
- Presentation of final NEMO meta-architecture. The updated version is based on feedback received from development and integration activities, as well as initial pilot results. The final architecture realizes slight updates related to the network, user, operational and physical view. It refines the organization of managed resources into clusters, with regards to their placement in the continuum. Greater updates are provided for the logical view, which has been aligned with developments, while the process view specifies additional metaOS processes or at finer details. Moreover, the physical view provides instantiation of metaOS architecture in physical setups.
- The impact of the NEMO architecture and placement within the European metaOS landscape is discussed. This includes functional coverage of the NEMO metaOS, as well as extensions enabled through the architecture. Indicatively, the integration of Data Spaces is presented as a potential extension. In addition, the NEMO architecture is mapped to the reference architecture derived from the Architecture task force of the EUCloudEdgeIoT community. NEMO alignment is discussed at functional and components' level.
- Final specifications of the Validation & Verification (V&V) methodology of NEMO are presented. Specific tools are identified and suggested for V&V implementation in WP4.

The final architectural specifications will drive the development, integration and validation activities of the NEMO framework. Moreover, third-party integration, either as developers extending the architecture or consumers hosting their workloads into NEMO, is guided by the NEMO architecture, its approach for extensions and workload hosting.

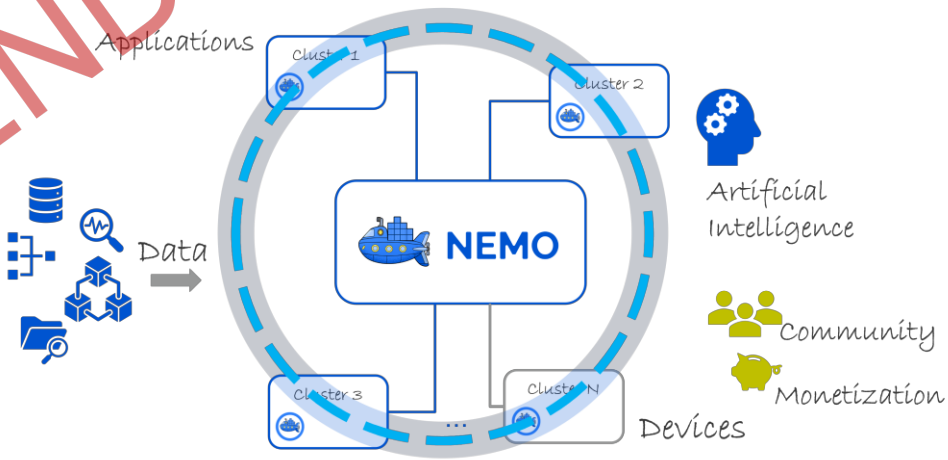
1 Introduction

The future of the digital world is modular and containerized. As digitalization penetrates with increasing rates in our personal and professional activities, myriads of software pieces arise to complete the puzzle of our digital well-being, supporting use cases hard to even imagine in the recent past and which demand their housing in the digital grounds of the connected world. As technology evolves, newer and more devices spring up, and uninterrupted online presence becomes a necessity.

Cloud provides virtually unlimited resources and high availability is considered a commodity. Inherent cloud features, like elasticity and multitenancy, have allowed production-level delivery of myriads of services, while providing distinct user's personal space (tenant). However, advances in technologies, like Augmented/Virtual/Mixed Reality, as well as advances in image/video resolution (reaching 8K to date) are just examples that have led to the emergence of applications demanding ultra-low latency. In addition, the increasing penetration of Artificial Intelligence (AI) has created stiff requirements on availability and access to large data volumes, raising privacy concerns. Here is where computation and intelligence at the 'edge' came into play and together with advances and enhancements in communication technologies through 5G & 6G embrace private/local computation and networking, which aspire to connect seamlessly to various clouds.

This would possibly not have happened without cloud native applications and services. Microservices' based architectures, i.e. independent, loosely coupled software modules providing small functional pieces together with the cloud and workloads' containerization have leveraged the flexible use of resources while delivering high-QoS (Quality of Service) and high-QoE (Quality of Experience) services since the cloud-only computing era. Those distinct software pieces, the microservices, may run at different points (nodes) across clouds and more recently across edges and even IoT devices. Depending on their design, microservices usually need to communicate in order to form/deliver together a greater, coordinated application logic.

Beyond cloud integration, NEMO aspires to meet develop a *meta-Operating System (metaOS)*, which will democratize the usage of resources anywhere in the IoT, edge and cloud continuum, while enabling the delivery of innovative microservices based applications of diverse capabilities' requirements through any device at everyone's hand. This vision aims to be materialized in NEMO, without compromising cybersecurity, trust and privacy requirements, while enabling business development and prosperity on top of this innovative ecosystem.



The NEMO *meta-Operating System* (metaOS) will enable multi-cluster and multi-network orchestration of containerized workloads across the IoT, edge and cloud continuum. As a (meta-)OS, NEMO will be user-centric, facilitating users to develop and deploy on top of NEMO. Moreover, NEMO will enable cloud and infrastructure providers to integrate their computing and networking resources into NEMO's infrastructure.

1.1 Purpose of the document

The present document is the third deliverable of Work Package (WP) 1 (D1.3) and reports the updates of the activities of Task 1.1 "Requirements analysis, use case refinement and target KPIs", Task 1.2 "NEMO meta-architecture design and components specifications" and Task 1.3 "Benchmarking definition and GDPR/Ethical compliance". Specifically, D1.3 reports the outcome of these activities which contribute to meeting the following WP1 objectives:

- Analyze the challenges, define the requirements and specification of the NEMO meta-Architecture.
- Produce the test reports format, parameter, test points and benchmarking for a unified and reliable outcome.
- Provide continuous technology monitoring on next generation IoT advancements and alignment with NEMO.

This document aims to provide the final version of the NEMO metaOS meta-architecture, which will materialize the NEMO vision and builds upon the first version defined in D1.2 "NEMO meta-architecture, components and benchmarking. Initial version" [1].

Moreover, the present document provides the final updates on the methodology adopted for NEMO Validation & Verification (V&V) benchmarking framework, guiding the V&V activities within WP4.

1.2 Relation to other project work

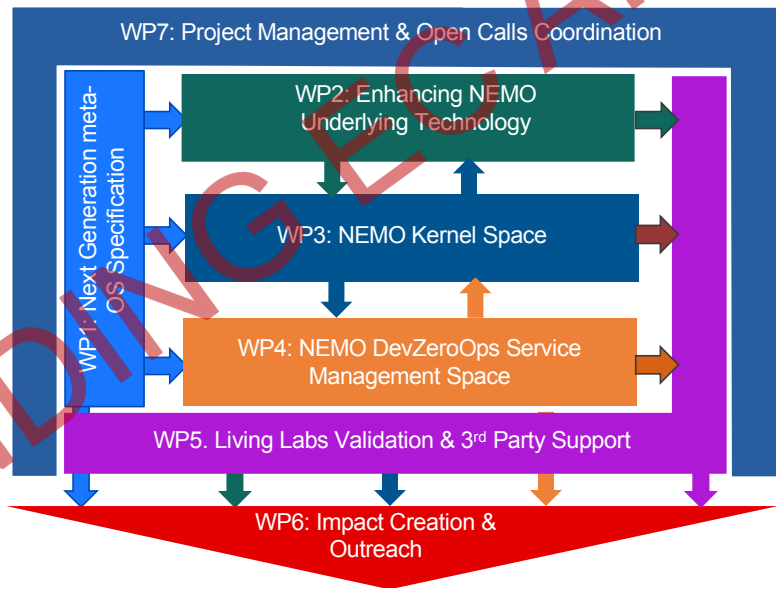


Figure 1: Feedback and interactions of NEMO activities (PERT chart)

The NEMO activities for building the envisioned metaOS are organized in work packages which interact with each other as depicted in Figure 1. Within Work Package (WP) 1 *Next-Generation metaOS specification,* the project identifies metaOS and IoT-Edge-Cloud continuum requirements, refines the use cases and defines its novel metaOS architecture. The specified user-driven requirements and technical design feed the activities in WP2, WP3 and WP4 for *Enhancing NEMO Underlying Technology*, the *NEMO Kernel Space* and the *NEMO DevZeroOps Service Management Space,* respectively. These include development of the NEMO components, integration and prototyping, as well as lab validation. Validated platform prototypes are provided to the living trials for pilot validation in WP5. Both lab and pilot validation are conducted against the metaOS requirements and results are evaluated through the project-defined measurable KPIs. Based on the outcome of this evaluation, new cycles of activities are triggered, aiming to enhance and fine-grain the prototype towards achieving the KPI targets.

1.3 Structure of the document

The rest of this paper is organized as follows. Section 2 presents the application sectors and use cases within the NEMO Living Labs. Section 3 describes the contributions of the project in terms of architectural specifications that could act as reference or architectural patterns guiding development of metaOS solutions. Section 4 describes the V&V methodology for NEMO and how it will be applied both for NEMO components and third-party NEMO hosted workloads.

2 NEMO Use Cases & Requirements' Update

2.1 Overview

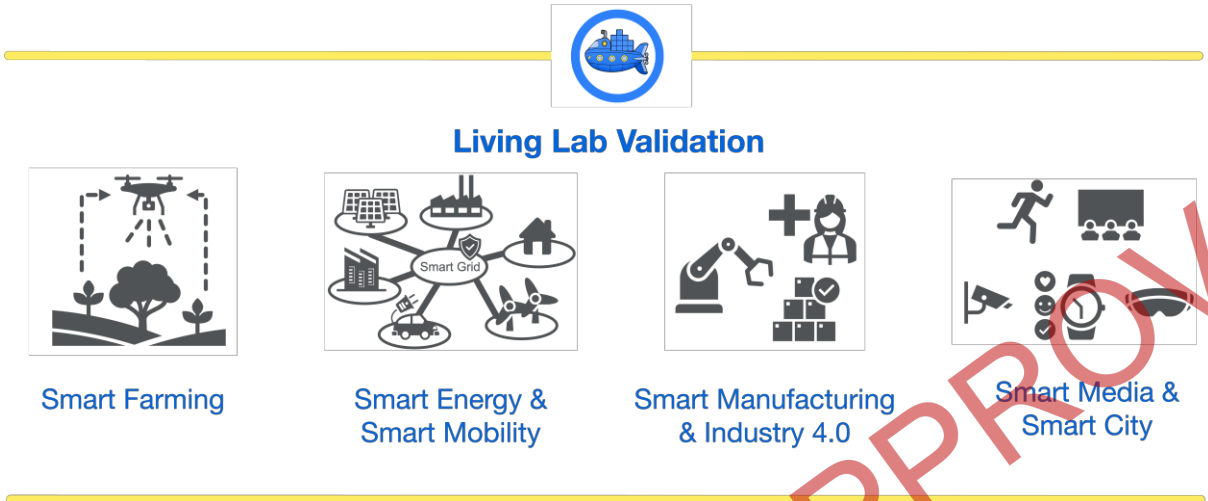
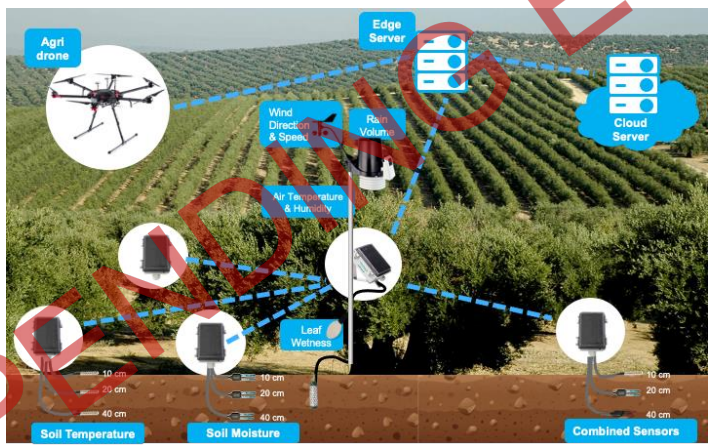


Figure 2: Application domains covered by NEMO Living Labs

2.2 Refined list of requirements

2.2.1 Smart Farming



Location

Monemvasia, Greece

Objective

Protection of olive trees from olive fruit fly through aerial spraying

Optimization of the use of bio-spraying, without compromising organic certification

Efficient and responsible resource utilization within the Smart Farm

Partners

ENTERSOFT OTE SYNELIXIS

2.2.1.1 Brief description

In the era of intense environmental pollution and draining of resources worldwide, availability of sufficient and high-quality food supplies is greatly threatened, while over-cultivation combined with excessive use of chemicals as either fertilizers or pesticides even endanger food safety.As a noble alternative, organic farming promotes sustainable and environmentally friendly agricultural practices. Olive cultivation plays a crucial role in numerous economies worldwide, requiring efficient management practices for optimal yield and sustainability. Olive oil production is greatly affected by the climate change of the recent years realized intensely in major European olive producing countries, like Spain, Italy and Greece.

On the other hand, olive fruit fly development is favoured by the lack of heat waves and rain. The olive fruit fly (scientific name "Bactrocera oleae" or "Dacus oleae (Gmelin)") is a major concern for olive tree cultivation, causing serious damage to the olive fruit, significantly affecting both quantity and

quality of yields [2]. Kaolin clay or spinosad-based bait are often used for olive fruit fly control as an organic treatment and have been used to protect plants from various insect pests, achieving protection against total and harmful fly infestation [3]. The Smart Farming Trial addresses organic farming in olive groves.

Precision spraying allows targeted application of pesticides and fertilizers, minimizing waste and environmental impact [4]. Unmanned Aerial Vehicles (UAVs) are already widely used for precision agriculture, offering rapid and detailed data collection capabilities [5], [6] and field monitoring using captured images. Leveraging on UAV capabilities and machine learning based innovations, autonomous precision spraying may lead to a good balance between organic olive yield quantity, quality and costs through collaborative intelligent systems. Moreover, Autonomous Guided Land Vehicles (AGLV) have largely contributed to smart agriculture by automating tasks that traditionally required significant human labour in precision farming (e.g. planting, fertilizing) [7], crop monitoring [8] [9], weed and pest control [10] [11].

The NEMO potential in enhancing organic farming will be validated in the Smart Farming trial, which deals with organic olive tree cultivation. The NEMO trial involves two use cases which address pest management and weed control respecting organic farming. Through NEMO, we aim to mobilise stakeholders in Smart Agriculture to cooperate through technological advances in IoT, edge/cloud computing and AI for precision agriculture, aiming to advance the level of automation and minimize human intervention, increase the quality of agricultural operations, as well as the quality and quantity of yields, while minimizing the technological footprint of agricultural processes.

2.2.1.2 Use cases update

The NEMO capabilities are validated in the Smart Farming domain, through two use cases, as defined in D1.1 [12].

The SF_01 Aerial Precision Bio-Spraying use case aims to protect the olive trees from olive fruit fly through aerial spraying conducting by UAV. This use case combines microclimate data collected via Synelixis SynField® [4] IoT nodes and real-time video analysis of olive groves from visual and multi-spectral cameras attached on semi-autonomous drones to identify in real-time where bio-spraying is needed. The bio-spraying decision will be based on ML models, which will optionally run on the end devices (UAV) or edge devices. Increased model performance and increased energy efficiency will be investigated during the training process through Cybersecure Federated Deep Reinforcement Learning (CF-DRL) and flexible deployment of the training jobs across the IoT, edge and cloud resources available.

The SF_02 Terrestrial weed management use case aims to organically control weeds in olive groves through Autonomous Guided Land Vehicles (AGLV). Autonomous robots equipped with cameras and sensors collect data for detecting obstacles and enabling autonomous weed mowing. The use case relies on ML models for the detection of obstacles like trees and humans, as well as for the autonomous movement within the olive grove. The ML services may run on the AGLV or at the edge resources of the Smart Farm or even move to servers of another farm for the sake of energy efficiency or ensuring high Quality of Service.

2.2.1.3 Requirements update

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
SF_01_FR01	The platform must provide access to measurements.	M	API-Access Control-MO-mNCC	80%	NEMO should ensure that authenticated/authorised users can access relevant endpoints through DNS names. This requires relevant support and integration of the NEMO Intent-Based API, Access Control, the Meta-Orchestrator, and the meta-network Cluster Controller.
SF_01_FR02	The platform must provide options to manage/view sensors/devices.	M	MO	100%	Devices operating as NEMO resources are integrated as nodes in clusters managed by the Meta-Orchestrator, which provides interfaces to manage the relevant nodes based on relevant roles and permissions.
SF_01_FR03	The platform must provide options to manage users.	M	IAM	90%	NEMO integrates Keycloak as the Identity Management components, ensuring single sign-on (SSO) and Role-based Access Control (RBAC) throughout the NEMO platform.
SF_01_FR04	The platform should support ML/FL training and ML model sharing/serving.	M	CFDRL	80%	CFDRL supports ML operations for NEMO, including ML model training, storage, serving and sharing. ML training is supported through federated reinforcement learning and privacy-preserving federated learning.
SF_01_FR05	The platform should provide ML classification accuracy probability.	S	CFDRL	90%	CFDRL modules for ML training provide this information after the training process completes.
SF_01_FR06	The platform should support automated aerial spraying.	S	API-MO-mNCC	80%	NEMO can accept workloads which support the aerial spraying operations (e.g. developed in T5.2). These workloads can be registered and deployed through the Intent-based API, the Meta-Orchestrator and mNCC, with the latter two responsible for their execution under the declared requirements on the container and network management side, respectively.
SF_01_FR07	The platform should support	S	PPEF	100%	Energy consumption metrics are already included and monitored

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
	monitoring of SLOs, e.g. related to energy consumption or CO2 emissions.				by PPEF, for different levels of abstraction (e.g. device, cluster, workload).
SF_01_FR08	The platform must respect data sovereignty and privacy requirements.	M	PPEF & Access Control	90%	PPEF adopts the PEP (Policy Enforcement Point) design pattern, including identity management and RBAC in the PDP (Policy Decision Point), ensuring that access to data (communication endpoints) is controlled by relevant authentication and authorization. In addition, Workloads owners may select the resources (clusters) where their workloads will be executed, and MO ensures that their execution is restricted in the said clusters. In SF use cases, SF application providers may select to execute their workloads in their own or their partners' resources, integrated as managed clusters in NEMO.
SF_01_FR09	The platform must support collection of monitoring data, such as the weather and plant conditions.	M	API-MO-mNCC	80%	The MO and mNCC support the execution of microservices-based workloads, which may run at different levels (IoT, edge, cloud) and interconnect with each other. As such, the workloads referring to the collection and communication of relevant measurements are supported under the declared requirements.
SF_01_FR010	The platform must support retrieving photos via drones.	M	API-MO-mNCC	80%	NEMO can accept workloads which support photo capture and retrieval (e.g. developed in T5.2). These workloads can be registered and deployed through the Intent-based API, the Meta-Orchestrator and mNCC, with the latter two responsible for their execution under the declared requirements on the container and network management side, respectively.
SF_01_FR011	The monitoring devices must	M	API-mNCC	80%	mNCC caters for the interconnection of microservices which may run in the same or

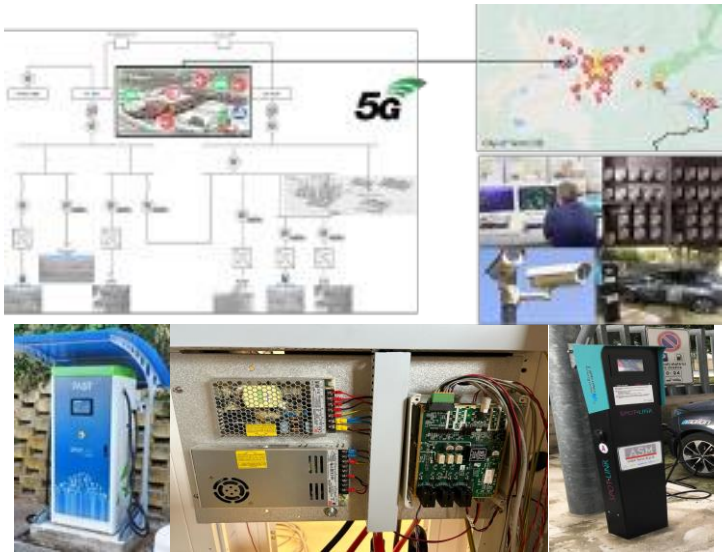
FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
	support network connectivity.				different clusters, even across administrative domains. The devices used in the SF trial have the required (hardware) capabilities to support this connectivity and integration to NEMO clusters.
SF_01_FR012	The monitoring devices must be able to communicate data to and receive control commands from the NEMO platform.	M	LCM / NEMO plugins	90%	FIWARE Context Broker has been employed as a NEMO plugin in order to allow communication of both data and control commands.
SF_01_FR013	The platform should be able to perform alternative scheduling or geographical distribution of smart farming services based on user goals.	S	MO-CFDRL	70%	The MO supports scaling and migration of workloads. MO decision on such actions relies on intelligent models regarding the placement and scaling of resources, which are provided by CFDRL.
SF_01_FR014	The Smart Farmer should be able to define strategies for the use of available resources.	S	API, LCM	90%	Smart Farmers (workload owners) can declare their requirements (through intents) for executing their workloads (SF applications). These may include several properties, e.g. referring to computational resources, energy or even accounting cost.
SF_02_FR01	Mobile robots must support autonomous operation.	M	API-MO-mNCC	80%	NEMO can accept workloads supporting mobile robots' autonomous operation (e.g. developed in T5.2). These workloads can be registered and deployed through the Intent-based API, the Meta-Orchestrator and mNCC, with the latter two responsible for their execution under the declared requirements on the container and network management side, respectively.
SF_02_FR02-**deprecated**	Mobile robots must be able to understand the weeds they should spray.	M			The use case has been reconsidered to apply weed mowing, instead of spraying, in order to comply with organic farming practices.

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
SF_02_FR03	Mobile robots must be able to follow a route or reach a destination.	M	API-MO-mNCC	80%	NEMO can accept workloads supporting mobile robots' autonomous operation (e.g. developed in T5.2). These workloads can be registered and deployed through the Intent-based API, the Meta-Orchestrator and mNCC, with the latter two responsible for their execution under the declared requirements on the container and network management side, respectively.
SF_02_FR04	Mobile robots must support network connectivity, such as Wi-Fi or cellular.	M	API-mNCC	80%	mNCC caters for the interconnection of microservices which may run in the same or different clusters, even across administrative domains. The devices used in the SF trial have the required (hardware) capabilities to support this connectivity and integration to NEMO clusters.
SF_02_FR05	The platform must respect data sovereignty and integrity.	M	PPEF & Access Control	90%	[Same as SF_01_FR08]
SF_02_FR06	The platform must provide access to collected data.	M	API-Access Control-MO-mNCC	80%	[Same as SF_01_FR01]
SF_02_FR07	The platform must provide access to the devices.	M	MO	100	[Same as SF_01_FR02]
SF_02_FR08	The platform must provide options to manage users.	M	IAM	90%	[Same as SF_01_FR03]
SF_02_FR09	The platform must be able to calculate mobile robots' routes in real-time.	M	API-MO-mNCC	80%	NEMO can accept workloads supporting the route (re-)calculation of mobile robots (e.g., developed in T5.2). These workloads can be registered and deployed through the Intent-based API, the Meta-Orchestrator and mNCC, with the latter two responsible for their execution under the declared requirements on the container and network management side, respectively.

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
SF_02_FR010	The platform must be able to identify and avoid obstacles such as humans and trees in real-time.	M	API-MO-mNCC	80%	NEMO can accept workloads supporting obstacle detection (e.g. developed in T5.2). These workloads can be registered and deployed through the Intent-based API, the Meta-Orchestrator and mNCC, with the latter two responsible for their execution under the declared requirements on the container and network management side, respectively.
SF_02_FR011	Mobile robots must be able to provide data to and receive control commands from the NEMO platform.	M	LCM /NEMO plugins	90%	[Same as SF_01_FR012]
SF_02_FR012	The Smart Farmer should be able to define strategies for the use of available resources.	S	API, LCM	90%	[Same as SF_01_FR014]
SF_01_NFR01	The NEMO platform must respect security and privacy requirements.	M	PPEF, IAM, NIM & Access Control	90%	NEMO incorporates a set of security and privacy modules, including IAM, Network Intercommunication Module, PPEF, Access Control, while adopting security practices at possible levels (e.g., CFDRL, SEE, CMDT, MOCA to protect components and data from unauthorized access and control.
SF_01_NFR02	NEMO should support High Availability features.	S	MO	100%	With horizontal scaling even across clusters and administrative domains, the MO supports high availability for NEMO hosted workloads.
SF_01_NFR03	The Smart Agriculture Application of NEMO should be vendor-independent	S	MO-LCM (plugins)	100%	NEMO supports the integration of diverse devices (far-/near edge, cloud) as nodes in NEMO managed clusters. Also, FIWARE Context Broker is employed as a NEMO plugin to allow communication of data and control messages with IoT devices. Other plugins can be employed to allow integration of

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
					generated data streams with Data Spaces. This allows SF application developers to adopt established communication protocols and thus being able to integrate devices from various manufacturers.
SF_01_NFR04	The NEMO platform should be scalable in the sense of providing additional resources when computationally heavy tasks are initiated.	S	MO-CFDRL	70%	Autoscaling decision support is developed in CFDRL to guide the relevant scaling actions by the MO.
SF_02_NFR01	The NEMO platform must respect security and privacy requirements.	M	PPEF, IAM, NIM & Access Control	90%	[Same as SF_01_NFR01]
SF_02_NFR02	NEMO should support High Availability features.	S	MO	100%	[Same as SF_01_NFR02]
SF_02_NFR03	NEMO should be flexible and scalable in the sense of exploiting available resources according to set goals.	S	MO, mNCC, PPEF, CFDRL	70%	Combining the capabilities of MO, mNCC, PPEF and CFDRL, NEMO will be able to intelligently manage container and network operation for the execution of workloads, aiming to fulfil user declared intents.

2.2.2 Smart Energy & Smart Mobility



Location

Terni, Italy

Objective

Monitor and stabilize the electricity smart grid

Improve Renewable Energy Sources (RES) load balancing via EV chargers

Predict traffic flow/parking prediction via EV chargers and parking positions for Mobility.

Support citizens eco-mobility in a smart city scenario combining crowd sourcing info and public transportation, weather/noise data, along with historical data and analysis of CCTV/traffic

Partners

2.2.2.1 Brief description

In the face of escalating environmental challenges and resource depletion, the stability and efficiency of urban energy ecosystems are significant. The Smart Energy & smart Mobility (SEM) pilot in Terni (Italy) is a pioneering initiative aimed to enhance grid stability, optimize renewable energy integration, and foster sustainable urban mobility. The pilot leverages an advanced infrastructure comprising 4 Medium/Low Voltage substations, a 200kW photovoltaic plant, 65 smart Electric Vehicle (EV) chargers, 10 Power Quality Analyzers, 2 Phasor Measurement Units (PMUs), and 100 smart meters. Additionally, the pilot incorporates a fleet of six leased EVs and at least 3 more smart EV chargers, creating a robust framework for smart city mobility.

2.2.2.2 Use cases update

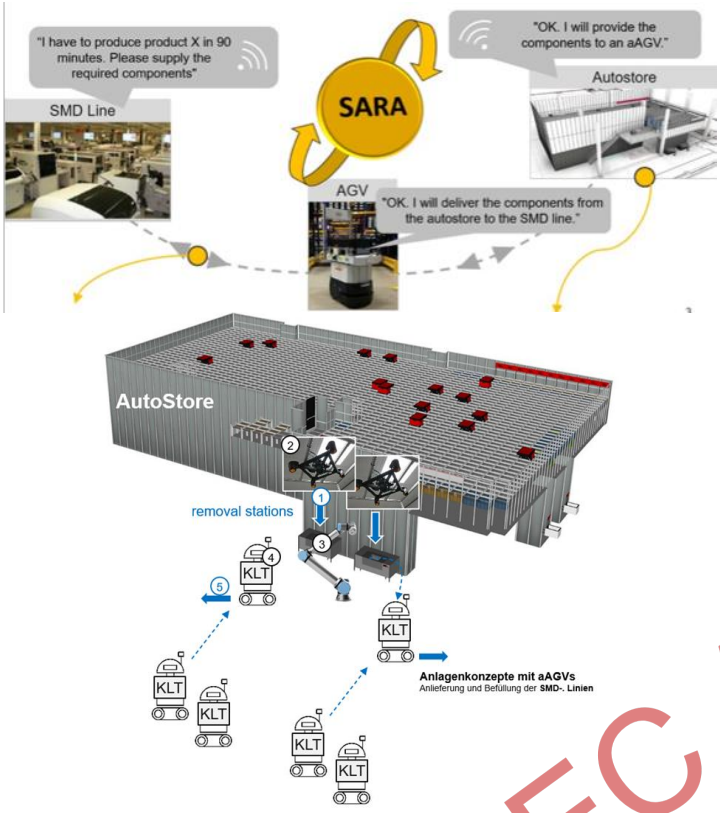
The NEMO capabilities are validated in the Smart Energy & Smart Mobility (SEM) domain, through three use cases:

- SEM_1 **Hierarchical Grid Disturbance Mitigation:** Observes the MV grid segment in real-time utilizing the PMU infrastructure. It complements PMU with edge gateway/computing for data pre-processing and disturbance event detection. Upon local event detection, it triggers disturbance localization and classification procedures. This provides the operator with insights to make informed decisions.
- SEM_2 **Flexibility Assessment and Valorisation:** Utilize the infrastructure of power quality meters to build a generic MV substation analytic model. Utilize the model in the transfer learning framework, leveraging the pre-training and knowledge for fast and efficient adaptation. Based on contextual information about assets connected to specific substations, the flexibility potential can be assessed by disaggregating the local generation, inflexible baseline, and flexible baseline. Optimize flexible assets' schedules and thereby support the grid.
- SEM_3 **Crowdsourced Smart Grid, Parking, and EV Charging Coordination** Improve Renewable Energy Sources (RES) load balancing via EV chargers. Predict traffic flow/parking prediction via EV chargers and parking positions. Support citizens eco-mobility in a smart city scenario combining crowd sourcing info and public transportation, weather/noise data, along with historical data and analysis of CCTV/traffic.

2.2.2.3 Requirements update

FR ID	Description	Component	Status (%compl.)
SEM_1_FR1	The platform must provide access and connectivity to edge and far-edge devices.	MO, SEE	50%
SEM_1_FR2	The platform must provide a way to gather and store data from the edge and far-edge devices.	PPEF, mNCC	30%
SEM_1_FR3	The platform must provide analytics capabilities for later decision-making process.	PPEF, CFDRL	50%
SEM_1_FR4	The platform should support ML training and ML model serving.	CFDRL	80%
SEM_1_NFR1	The platform must provide horizontal and vertical migration along with seamless migration toward optimal location.	MO	0%
SEM_2_FR1	Exploratory analysis of available data.	/	40%
SEM_2_FR2	The platform must provide access and connectivity to edge and far-edge devices.	MO	0%
SEM_2_FR3	The platform must provide a way to gather and store data from the edge and far-edge devices.	PPEF, mNCC	40%
SEM_2_FR4	The platform must provide analytics capabilities for later decision-making process.	PPEF, CFDRL	50%
SEM_2_FR5	The platform should support ML training and ML model serving.	CFDRL	50%
SEM_2_NFR1	The platform must provide horizontal and vertical migration along with seamless migration toward optimal location.	MO	0%
SEM_3_FR1	Deployment of OBD modules, and access to smart charging stations.	/	20%
SEM_3_FR2	The platform must provide options to manage users and manage users' access to recommendations.	IdM	0%
SEM_3_FR3	The platform must provide access and connectivity to edge and far-edge devices.	MO	30%
SEM_3_FR4	The platform must provide a way to gather and store data from the edge and far-edge devices.	PPEF, mNCC	20%
SEM_3_FR5	The platform must provide analytics capabilities for later decision-making process.	PPEF, CFDRL	20%
SEM_3_FR6	The platform should support ML training and ML model serving.	CFDRL	20%
SEM_3_NFR1	The platform must provide horizontal and vertical migration along with seamless migration toward optimal location.	MO	0%

2.2.3 Smart Manufacturing & Industry 4.0



Location

Ingolstadt, Germany

Objective

Improve mass production and safety in factories with high levels of automation, enabling Collaborative Robot (Cobots) systems, Automated Guided Vehicles (AGVs) and humans to co-work.

High-speed heterogeneous connectivity using 5G NR, TSN and Wi-Fi and various types of AGVs.

Analyse input from sensors, 3D cameras and RFID nodes and predict, identify and avoid collisions between humans and AGVs and between different types of AGVs

Partners

Continental, SIMAVI Software Imagination & Vision, Telefónica, Cumucore

2.2.3.1 Brief description

The Smart Manufacturing & Industry 4.0 Pilot trail at Continental plant Ingolstadt considers two applications to be implemented, relevant to validating NEMO.

The SM_01 “Fully automated indoor logistics/supply chain” targets ADAS manufacturing. Currently, handling and transport of material (SMD Components) from the Auto Store to the production sites are performed manually every 30 minutes. By utilizing a 3D-Vision camera for Bin Picking Application, an integrated Barcode Scanner and collaboration between different robot systems, Continental aims to fully automate controlled material picking from Auto Store and autonomous transfer to the production line.

The SM_02 “Human-centered indoor factory environment safety” will provide a high precision AGV localization layer merging real time localizations info obtained from cognitive sensors (safety cameras, radar and Lidar). A high-speed and ultra-low latency (TSN) private wireless network will support massive data uploads to the edge cloud facilities, where AI functions will detect the position of each body and build a "safety shell" around it to ensure human-centred safety, while federated CF-DRL will enable model transfer learning to the AGVs to enable autonomous avoidance of potential collision between AGVs, or between a worker and an AGV.

2.2.3.2 Use cases update

The central process "Bin Picking / Pick and Place" in the Smart Industry 4.0 use case is recreated in STAR (Smart Technology & Application Room) to test the NEMO integration. The hardware setup for the demonstration has been completed. The following components and interfaces are used.

Devices / Tools	level	Communication / Interfaces
Camera		
• ActiNav (PhotoneoPhoXi3D Scanner)	Edge/Cloud	Ethernet
• DALSA GENIE NANO-5G-M4040 (Area Scan)		Ethernet
• SICK (PLB 5.0)		Ethernet
• Keyence CV-X480D		Ethernet
Robot		
• TM5	IoT/Edge	RS-232/RS-422/RS-485/Ethernet/Modbus TCP/RTU(master&slave) PROFINET/EtherNet/IP
• TM12		
• UR3		TCP/IP 1000 Mbit: IEEE 802.3u, 100BASE-T Ethernet-Buchse, Modbus-TCP & EtherNet/IP-Adapter, Profinet
• UR5e		
• RobCo		EtherCAT, Ethernet, Modbus TCP, EtherCAT, RJ45-Buchse

2.2.3.3 Requirements update

From a technological point of view, the Smart Manufacturing & Industry 4.0 Pilot aims to implement and validate the innovative architectural solution proposed by NEMO in order to fully automate the production workflows. The solution developed within the Pilot will achieve two major objectives at the level of process automation, namely:

- monitoring and detection during the manufacturing workflow
- and ensuring the manufacturing safety procedures

The NEMO platform will provide the functionalities of two monitoring systems through equipment and sensors, namely the Bin Picking system for the automation of the manufacturing process and the sensor system for assessing the working environment.

At the communications' infrastructure level, an IoT/5G Time Sensitive Networking system will be implemented, and the services will be orchestrated through intelligent Open API algorithms. The trial will emphasize on implementing next-generation IoT applications related to AGVs-AGVs and AGVs-human operators' collision prediction, detection and avoidance through real-time positioning and federated ML hosted locally and in the 5G/WiFi edge.

2.2.4 Smart Media & XR

Location

Athens, Greece

Objective

Validate NEMO for live event Smart Media/crowd sourcing applications (i.e. many users, live content annotation, high bandwidth 5G requirements)

Validate NEMO for eXtended Reality (XR) environments (i.e. high interactivity, low latency, accurate gesture/emotion analysis)

Partners

FOUNDATION OF THE HELLENIC WORLD
Maggioli
POLITÉCNICA
Telefónica
OTE
novomatix

2.2.4.1 Brief description

2.2.4.1.1 Smart Media City

This trial tries to enhance the live running sports' event spectating experience by enriching the content through AI-driven data and content analysis and XR capabilities. During the race, media content is captured by many spectators and selected runners along the running circuit using smartphones/tablets and GoPro cameras, and if existent IP cameras and drones.

Incoming content is automatically processed, annotated, and rendered (partially on the device using already trained AI/ML models and partially at the edge), and a selection is directly broadcasted (e.g. via social media) based on location info of the (top) runners and interesting events during the race (e.g. based on contributor annotation).

The audience has the option to improve their contributions and can interact with contributors in case of specific race incidents. The emphasis is on real-time user-generated content processing and rendering using FL hosted locally on the IoT nodes (smart phones), in the edge and at the cloud. The AI/Models will be trained to recognize the Racing Bib Numbers on each athlete and street numbers and landmarks of the race in order to first understand the runners in each stream and then enhance the positioning identification of the stream and runners in it.

2.2.4.1.2 Smart Media XR

The XR use cases will be hosted at the Hellenic Cosmos Cultural Centre of the Foundation of the Hellenic World. The Hellenic Cosmos is a multifunctional area where visitors experience Hellenic history and culture, while at the same time it is a venue of cultural creation and expression. In its areas we organize a wide range of activities, open to people of all ages and interests. The XR use cases will enhance two VR experiences that are available to the public using gestures recognition and biometric data. The experiences that will be enhanced are: a) A VR Head Mounted Display (HMD) experience regarding the visit of an ancient Greek Workshop and b) an interactive real-time VR Dome experience that is presented at the Tholos Dome VR Theatre of the Hellenic Cosmos. The trial consists of several micro-services that are going to be deployed in the continuum (central cloud, edge cloud, IoT devices) via the NEMO platform. The infrastructure in FHW premises (local VMs, servers, IoT/VR devices, etc) consists of the edge/IoT infrastructure and external infrastructures (Central Cloud, HPC servers) will be used for services that demand high computational resources.

In-depth description of both use cases can be found in deliverables D1.1 [12] and D5.2 [13].

2.2.4.2 Use cases update

2.2.4.2.1 Smart Media City

Figure 3: Smart City pilot infrastructure

This diagram in Figure 3 shows the pilot infrastructure. All the devices you see, are considered IoT devices and used for media capture. The smart phone is additionally used to consume the production feeds produced. It is evident that this use case will rely on a the 5G network to support the high bandwidth but also make heavy use of the NEMO mNCC, AI/ML components and intent based migration to support the real-time load. So, to summarize the benefits of this pilot:

- Fast and time sensitive migration across the continuum for large media.
- Media processing and AI/ML annotation nodes across the continuum to support multiple sources and users.
- Validate NEMO user acceptance from a citizen viewpoint for live sport events.

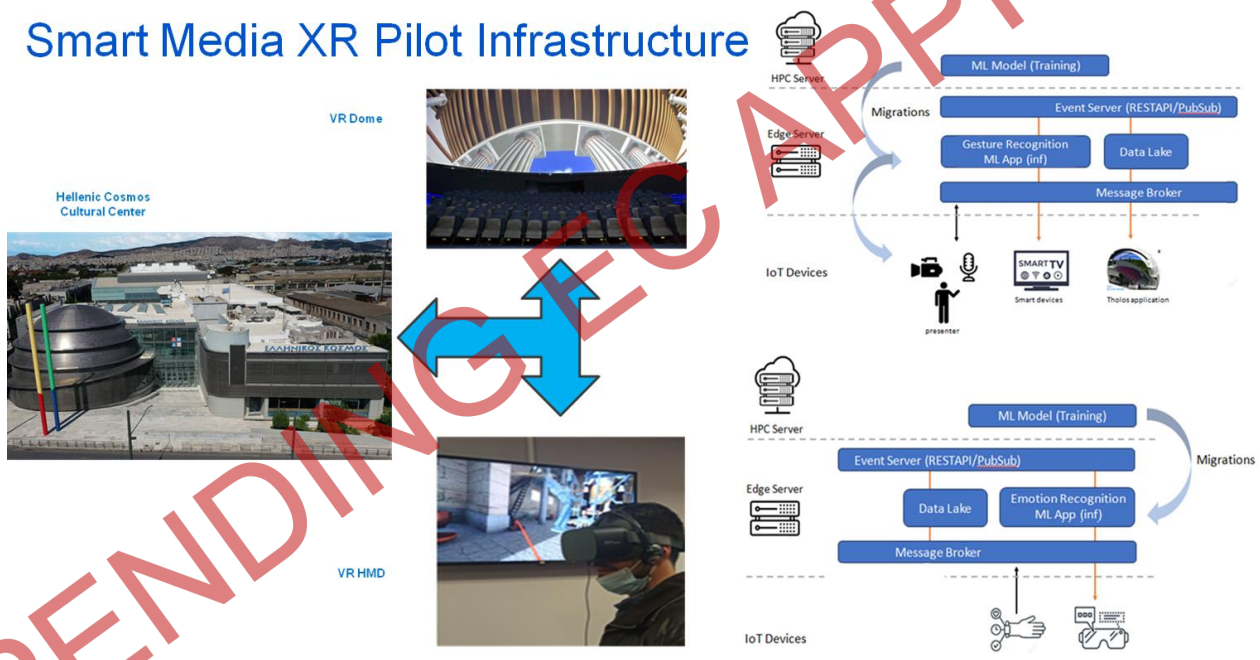
The NEMO capabilities are validated in the Smart Media City domain, through one use case, as defined in D1.1:

- SC_01 Enhance the live sports event spectating experience by enriching the content through AI-driven data and content analysis via NEMO components and manual production control. NEMO will provide the base components to facilitate communication, network access,

migration etc. This use case will make use of smartphones which are considered IoT devices that will be running a dedicated app. Media Gateway/Delivery managers for video content adaptation and delivery running on the cloud making heavy use of the mNCC, a media production engine which will be running on the cloud which will use NEMO resource management, an AI Engine running on the cloud running AI/ML annotations and intent based NEMO metaOS migrations to support the real-time load. All additional cameras are also considered IoT devices.

2.2.4.2.2 Smart Media XR

In trials we conducted in previous years we had the traditional approach of one CPU/GPU Node per user for enhancing an HMD experience or one high-performance node locally for enhancing the Dome. Each node needed a dedicated and measured network connectivity, manual installation and upgrade and training resulted in downtime. In contrast, NEMO promises lightweight low cost IoT devices, remote computing resources that feature automatic installation, migration and training, and way to allocate network resources on demand as well as providing a 24-hour uptime through its decentralized nature.



The NEMO capabilities are validated in the Smart Media XR domain, through one use case, as defined in D1.1:

- XR_01 VR Experience about ancient workshop of sculptor Phidias. Enhance experience with biometric data.
- XR_02 Enhance AV experience in the Tholos Dome VR Theatre. Analyse gesture of museum-educator presenter. Gesture based recognition by using ML in IoT-to-Edge-to-Cloud continuum.

These use cases will make use of local XR IoT devices, ML/AI components running on the edge with intent based migration to the cloud for training purposes and to support the real-time load, as well as communication components event servers, information storage and message brokers that will be supported by the Nemo mNCC and resource management.

2.2.4.3 Requirements update

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
SC_01_FR01	The contributors (real spectators) must be authenticated to access the tools and set up a transmission.	M	IAM-AC	10	User App will be entry point
SC_01_FR02	The broadcaster must be able to define/adjust the media-specific requirements of the transmission.	M	API - mNCC	10	Media managers will implement this
SC_01_FR03	The broadcaster must be able to monitor the signal quality and QoE parameters of the transmission to ensure streaming quality.	M	mNCC	10	Media Manager component
SC_01_FR04	Several video streams are to be transferred through the cloud/network. Bandwidth requirements must be met accordingly.	M	mNCC - MO	10	Media Processing engine
SC_01_FR05	Control signals (voice and data) and audio/video return channels are to be transferred between the technical director location and the venue via the cloud network.	M	API - mNCC	10	Media Processing Engine and Production Control
SC_01_FR06	NEMO must provide the adequate resources to the service provider to map these requirements onto the cloud network and perform accordingly.	M	IAS - LCM	10	Nemo OS specific OTE will host the cloud services
SC_01_FR07	The processing of the video streams is achieved automatically by the virtualised compression functions that are part of the Media Production Engine, which are deployed at the edge cloud near the venue.	S	CMDT - IMC	10	Media Processing Engine -Video Quality Probe
SC_01_FR08	NEMO will be able to allocate and launch the required services/VNFs on a location basis.	M	PPEF	10	Nemo OS specific for Media Managers
SC_01_FR09	The service provider must be able to chain services/VNFs with the help of a service orchestrator.	M	MO-mNCC	10	Nemo OS specific for Media Managers

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
SC_01_FR10	NEMO applies a central control unit (Cognitive Network Optimization) that is used by the service provider to adjust/adapt the network dynamically according to the specific requirements and conditions.	S	mNCC	10	Video Probe and Network Probe
SC_01_FR11	NEMO must be able to monitor and control the network and ensure adherence to QoS levels (bandwidth, average bit rate, round trip delay).	M	PPEF	10	Network Probe
SC_01_FR12	The production of a final stream on-site is realised with the help of the MPE module at the cloud edge. The technical director remotely controls the signal switching at the MPE.	M	API-SEE	10	Production control component
SC_01_FR13	Cognitive Services module allows for enrichment of the audio/video stream with additional information like face recognition, image recognition, data fusion, etc.	M	API-CFDRL	10	AI Engine
SC_01_FR14	A media app on a smartphone can be used (by journalists and/or the audience) for acquisition and streaming of audio-visual content into the cloud and make it available for the selection from technical director.	S	API-SEE	10	App
SC_01_FR15	Max. end-to-end network latency (RTT) - It comprises the latency of the whole network path excluding end devices on-site (like the network gateway or HW video coder) <= 50 ms	S	SEE-MO	10	Media Managers
SC_01_FR16	Minimum end-to-end connection bandwidth (per stream). Listed here are pure video data rates, for resulting data rates on the network layer add 10 % overhead: - Uncompressed HD (1080i and 1080p): 1.5 Gbit/s and 3.0 Gbit/s; - JPEG2000: 100 Mbit/s;	M	API-SEE-AC	10	Media Managers

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
	- H.264/AVC-Intra: 25 Mbit/s; - H265/HEVC: 15 Mbit/s				
SC_01_FR17	Error-free/lossless transport of signals with max. packet-loss rate: <10^(-12)	S	mNCC	10	Media Managers
SC_01_FR18	Max. network jitter/packet delay variation (PDV) < 10 ms	S	mNCC	10	Media Managers
SC_01_FR19	Classification and prioritisation of audio-video-streams. Due to the high requirements on latency, jitter and bandwidth media streams have to be prioritised in the network.	S	MO	10	Media Managers
SC_01_FR20	Max latency of end-to-end signal transport (video, audio and control data) - it comprises the latency of the whole signal path including converting of end devices on-site and media-specific VNFs). · Maximum E2E latency one way for video and audio: <= 500 ms · Max. E2E latency for return video (one way): <= 500 ms (Typically uses less bandwidth because of low-resolution proxy transfer) · Max. end-to-end latency for intercom (if needed): <= 100 ms (according to ITU G.114)	S	IAS	10	Media Managers
SC_01_FR21	Synchronisation of video and audio signals. GPS Synchronisation of end devices (using black burst, tri-level sync and/or PTP).	M	IAS-LCM	10	Media Processing Engine

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
SC_01_FR22	The MEC platform and underlying NFVI is required to deploy and run all the needed VNFs. The estimated use of resources is: · High CPU power, preferably new processor generation (>= 96 cores). · 128 GB RAM · 1 TB Storage SSD · Multiple 10 Gbit/s and 1 Gbit/s interfaces. · GPU processing capability.	M	IAS-LCM	10	Media Managers
XR_01.FR01	Collect user's biometric data	M	API-SEE	30	Local IoT device
XR_01.FR02	ML model for physical and emotional status detection	M	IAS-API	40	Edge ML component
XR_01.FR03	The solution must have an Application server (REST API) Service for communication between system admin and AR/VR application and UI interface for specifying what events and data to send depending on the state.	S	API-mNCC	80	Event component on the Edge
XR_01.FR04	Enhance VR app to subscribe and handle events	M	VR App	25%	FHW will provide the applications. First version of event server ready.
XR_01.FR05	Network will support diverse devices (wearables, AR/VR headsets) with different performance (e.g., high throughput, low latency and massive connection densities)	M	mNCC	80	IoT device Local
XR_01_FR06	Interoperability with external systems (i.e. multi sensorial stimuli system)	M	MO-LCM	80	Event server local component
XR_01_FR07	The platform components involving direct interaction with the end-users should be quick to respond to the users' actions	S	MO	80	IoT devices

FR ID	Description	MoSCoW	Component	Status (%compl.)	Comments
SC_01_NFR01	The platform must provide mechanisms for security and data privacy	S	SEE	10	VR app IOT auth
SC_01_NFR02	The platform should support high availability deployments	S	IAS	10	Migration to cloud High Performance Computer (HPC)
SC_01_NFR03	Live migration should be done using microservices.	S	IAS	10	HPC - Data Lake component
SC_01_NFR04	The platform should detect of network faults or malfunctions before those have any drastic impact on its performance	S	mNCC	10	Edge service components
SC_01_NFR06	The use cases depend on 5G network availability at level of five-nines for all communications	S	mNCC	10	Between IoT-Edge-Cloud migration
XR_01_NFR01	The platform must provide mechanisms for security and data privacy	S	CMDT-SEE	10	Edge security
XR_01_NFR02	The platform should support high availability deployments	S	MO-IAS	10	Migration to cloud High Performance Computer (HPC)
XR_01_NFR03	Live migration should be done using microservices live migration	S	IAS	10	Migration to cloud High Performance Computer (HPC
XR_01_NFR04	The platform should detect of network faults or malfunctions before those have any drastic impact on its performance	S	MO	10	Edge service components
XR_01_NFR06	The use cases depend on 5G network availability at level of five-nines for all communications	S	mNCC	10	Edge service components

3 Final NEMO Meta-Architecture

NEMO vision for the metaOS as an enabler of user-centric multi-cluster and multi-network orchestration of containerized workloads across dispersed IoT, edge and cloud resources. As such, a metaOS is conceived as a platform orchestrating multiple heterogeneous systems (IoT systems, edge infrastructures, cloud platforms) and data streams across several layers (network, compute, service, application) and across application verticals.

The first step towards achieving this vision has been the definition of an architecture for the innovative concept of the metaOS. Given that NEMO introduces one of the first contributions towards metaOS definition, the project has identified the need to propose a meta-architecture for building meta-Operating Systems (metaOS). In NEMO, the meta-architecture is conceived as a Reference Architecture on top of existing reference architectures, which aims to provide guidance on evolving or creating new metaOS architectures.

As a means to specify the meta-architecture, NEMO defined the Meta-Architecture Framework (MAF). MAF follows the definitions of ISO/IEC/IEEE 42010 definitions for the *architecture framework*, slightly adapted to support the meta-architecture concept. ISO/IEC/IEEE 42010 defines *architecture framework* as "conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders" [14]. NEMO has adopted a slightly modified term **"meta-architecture framework", as "conventions, principles and practices for describing meta-architectures established within an ecosystem of various domains of application and/or community of stakeholders"**. The NEMO meta-architecture aims to serve as a basis for building metaOS reference architectures, facilitating entities' integration in the metaOS world.

NEMO MAF, which includes the following elements, as depicted in Figure 4:

- Rationale
- Entity of interest
- Stakeholders
- Stakeholders' perspective
- Concerns
- Viewpoints
- Cross-cutting functions

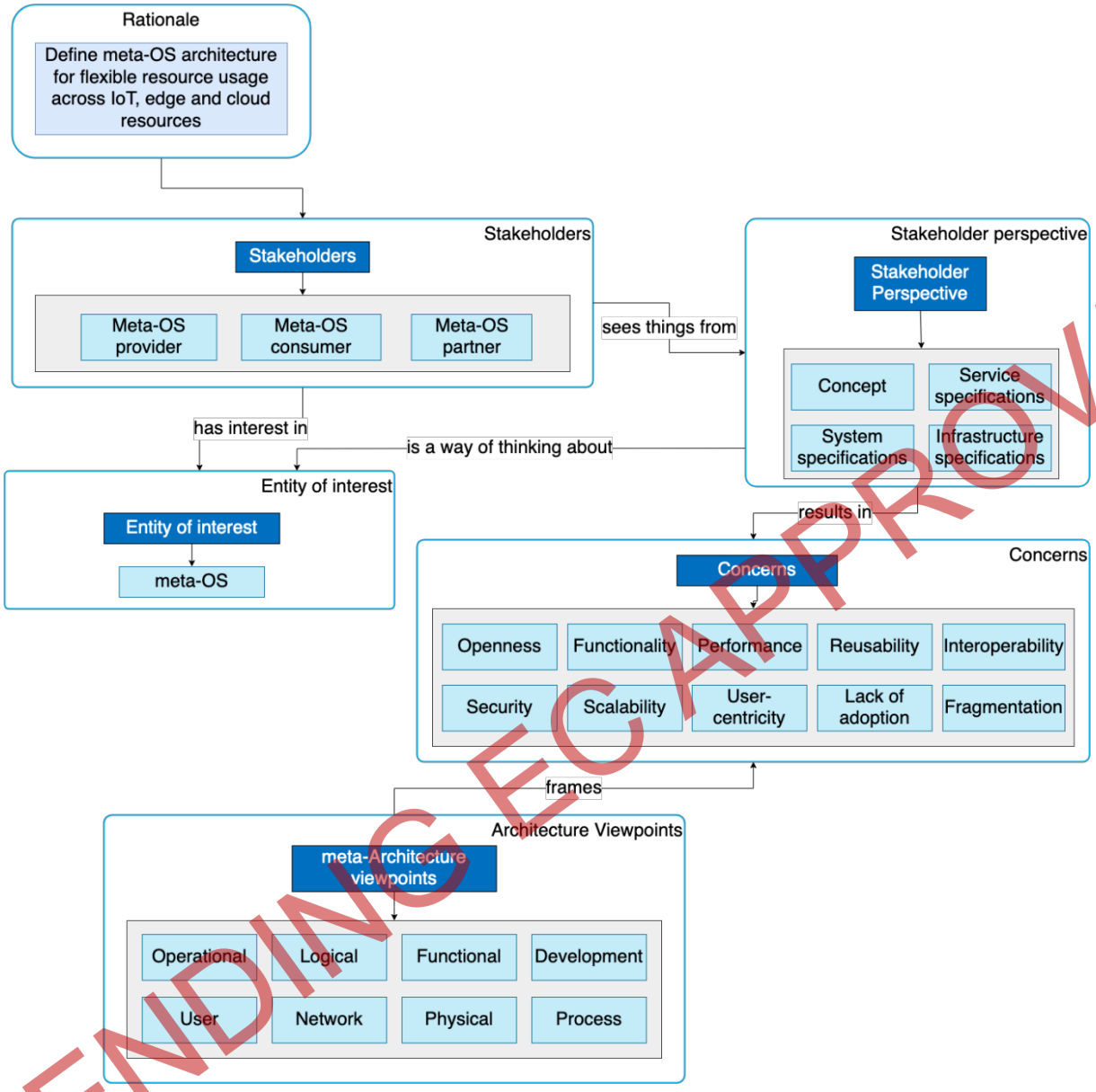


Figure 4: The NEMO metaOS Meta-Architecture Framework [1]

The NEMO metaOS MAF elements are described in detail in deliverable document D1.2 “NEMO meta-architecture, components and benchmarking. Initial version” [1].

3.1 Architectural Views

3.1.1 Network view

The Network view provides the physical and conceptual classification and hierarchy of the NEMO metaOS computing resources. Figure 5 provides the updated Network view of the NEMO architecture, which was originally defined in D1.2. Table 1 provides the definition of the network entities and concepts, considered in the NEMO metaOS.

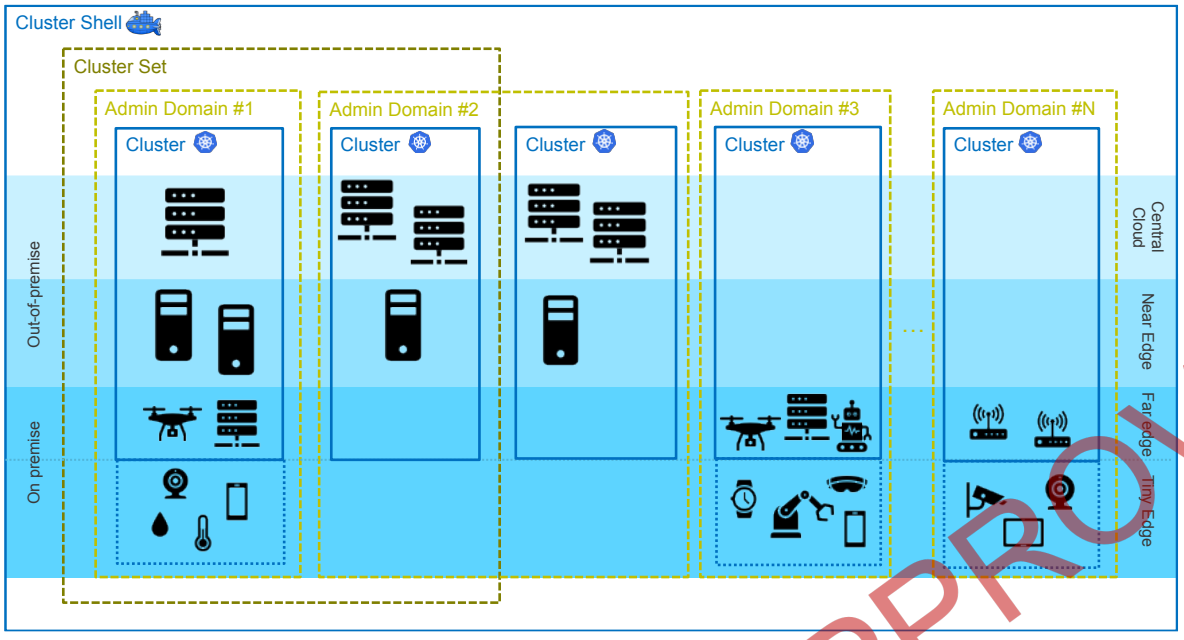


Figure 5: Network topology for the NEMO metaOS

Table 1: Definitions for NEMO network view

Network definition	Description
Tiny edge	Fixed-function devices, such as sensors, actuators and IP cameras within the enterprise network and infrastructure. As a result, it is a subgroup within the far edge, but the tiny edge devices have very constrained computing and storage resources.
Far edge	Nodes which are typically farthest from the cloud servers and belong to the IP space and infrastructure owned and managed at enterprise level.
Near edge	Devices which are *nearest* to the centralized services provided by cloud servers. Nodes in the near edge typically belong to the infrastructure hardware and IP space of communications service providers.
Central cloud	Server in micro-/datacentres with significant computing capacity, which could belong to any type of cloud, i.e. public, private, hybrid, etc.
Node	A node is a hardware computing element of the highest granularity in the metaOS continuum. As such, it may represent any physical computing device in the continuum, including IoT, Edge and Cloud Server Devices, or even a virtual machine provided by a cloud provider. As nodes are aimed to provide computing abstraction for executing containers of workloads on them, the classification into network levels is aimed to guide their configuration for allowing their integrated management through a common metaOS control plane, in a way that would be compliant to the computing capabilities of the node in terms of CPU, RAM and storage.
Cluster	A Cluster is defined as a grouping abstraction of nodes which are managed collectively by a centralized control plane. A cluster could include one or more nodes, either on- or out-of-premise, which altogether form the resource pool upon which different workloads' execution is orchestrated. Nodes can be added or removed from the cluster, without this affecting the execution of workloads. In NEMO, metaOS clusters are realized as managed K8s clusters.
Cluster Set	A Cluster Set provides a grouping abstraction for clusters participating in the metaOS. It allows the performance of workload management, policy

Network definition	Description
	enforcement, and resource sharing operations which apply to the clusters that are members of the Cluster Set. For example, the cluster set concept allows for addressing use cases which include *hybrid cloud management*, i.e. managing a mix of on-premises and cloud-based clusters under a unified policy framework. Also, it covers *multi-region deployments*, ensuring consistent configurations and policies across clusters in different geographic locations to reduce latency and improve user experience. Last, *disaster recovery*, scenarios can be supported, considering clusters within a cluster set for redundancy to implement disaster recovery plans by managing failover across clusters.
Cluster Shell	A Cluster Shell is the complete set of clusters participating in the metaOS. A Cluster Shell is scoped at the metaOS level and implies that clusters are managed in parallel in a coherent way through a unified control plane, covering hybrid and multi-cloud environments. The Cluster Shell allows performing cluster-aware administration, being aware and managing groups of clusters (i.e. Cluster Sets). It is composed of multiple clusters, allowing simplified management of available infrastructure and workloads, consistent policy enforcement, as well as enhanced scalability, high availability and resilience. Unless otherwise defined, a workload's execution should be able to be orchestrated in a coherent manner across all the clusters in a single metaOS instance.
Admin domain	The Admin domain refers to the IP space and infrastructure managed by a single entity, usually an enterprise, even if they do not belong to the entity. It represents the scope within which administrative control and governance are applied. An administrative domain is a cohesive boundary within which clusters, nodes, and associated resources are governed by a unified set of administrative policies, governance structures, and control mechanisms. It includes centralized control planes, policy enforcement, resource allocation, access management, and operational responsibilities, all managed by a designated administrative entity, which acts as a metaOS provider. Workload orchestration in the metaOS should be coherent across admin domains, respecting the security and privacy rules of the owning enterprise.

3.1.2 User view

The user roles identified in the NEMO metaOS are listed in Table 2. For each of these, sub-roles are defined in D1.2.

Table 2: User roles in NEMO MetaOS [1]

MetaOS User	Description
MetaOS Provider	This group represents parties that may host, provide and/or manage the metaOS.
MetaOS Consumer	This group represents consumers of the metaOS services, basically referring to application and service owners wishing to run their applications on the continuum.
MetaOS Partner	This group includes parties that may create value on top of the metaOS, which may result from integration of own resources, development on top of the metaOS, service brokerage and enablement, but also auditing.

3.1.3 Logical view

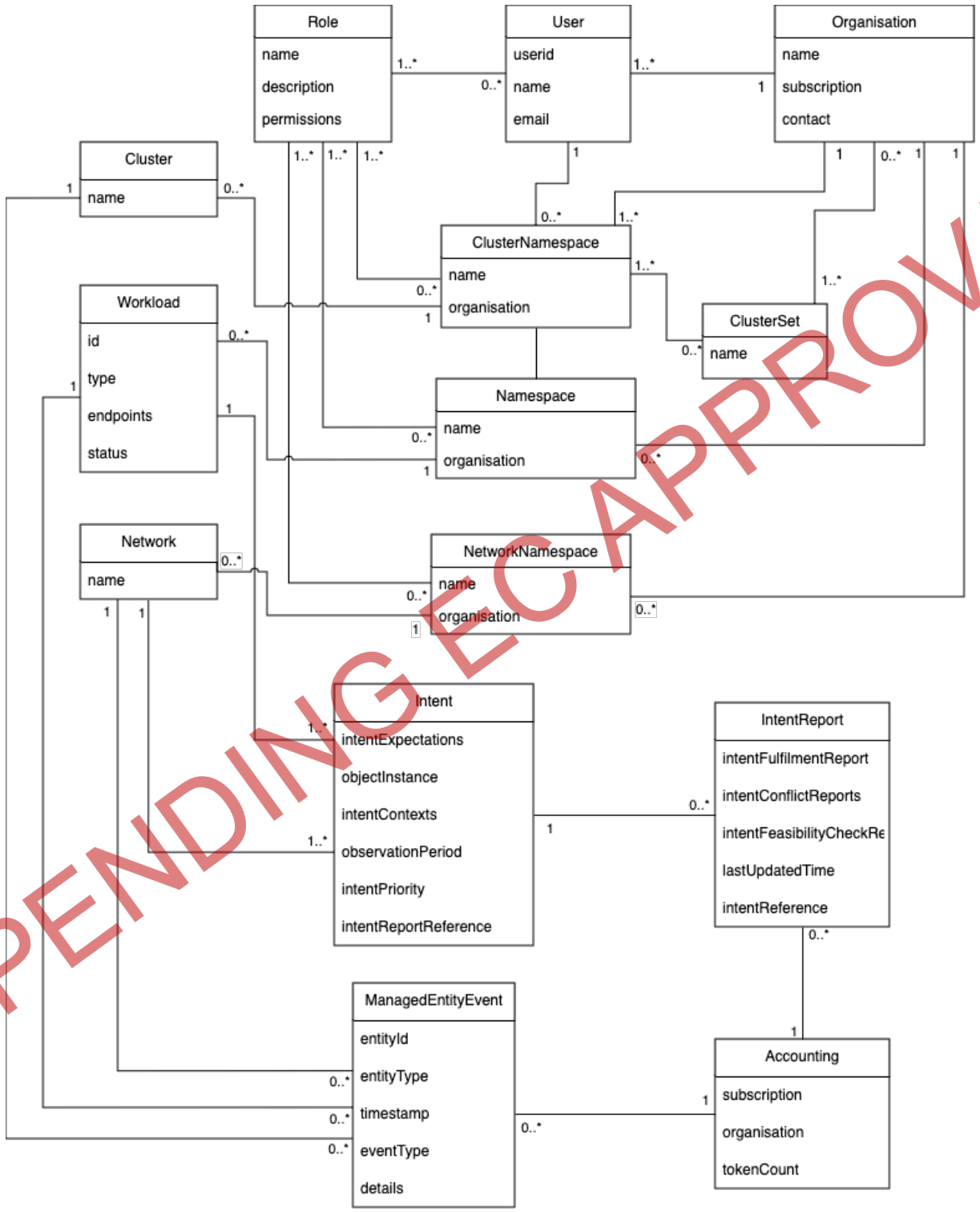


Figure 6: The logical view of the NEMO architecture

3.1.4 Operational view

The Operational View of the NEMO architecture is provided in the form of the NEMO Use Case scenarios and descriptions. The update of this view is provided in section 2.

3.1.5 Functional view

The functional view of the NEMO metaOS architecture is depicted in Figure 7. This view and functional layers have been defined in detail in D1.2 and have undergone no changes so far.

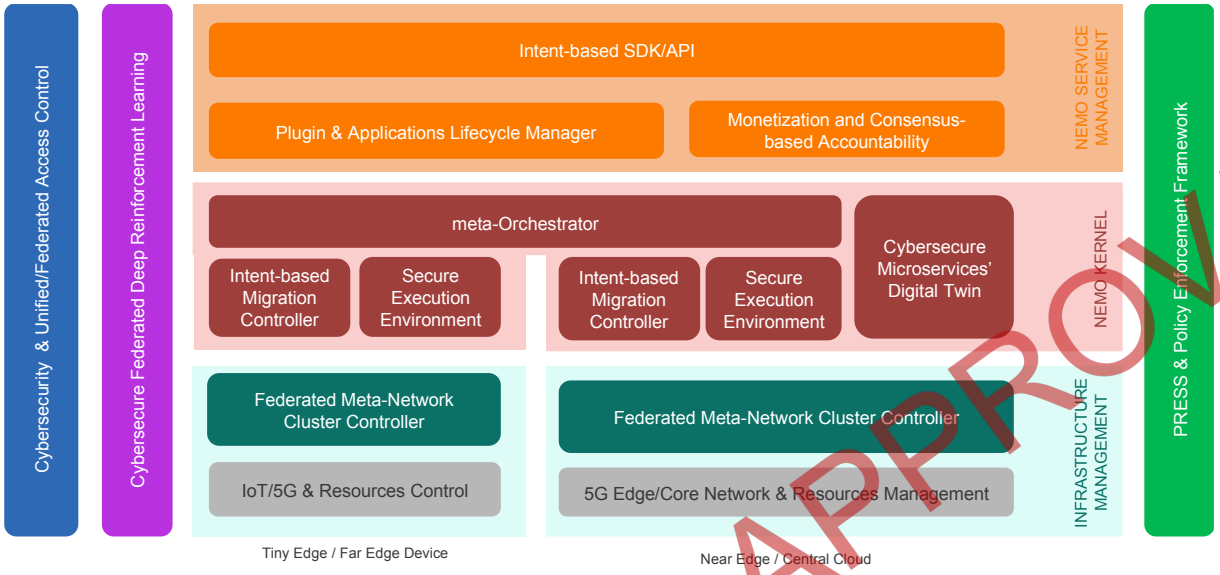


Figure 7: The functional view of the NEMO metaOS architecture

3.1.6 Process view

3.1.6.1 Resource provisioning

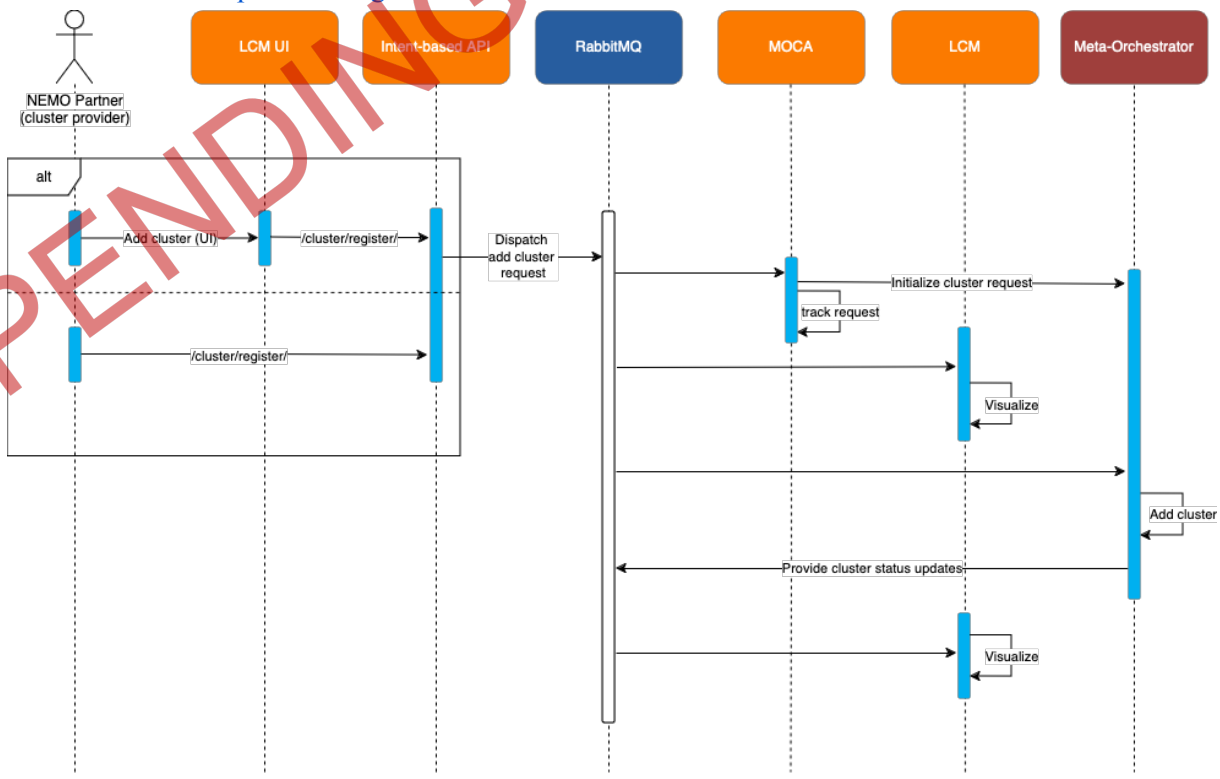


Figure 8: Process diagram for resource provisioning

Resource provisioning refers to integrating and onboarding additional infrastructure into the NEMO metaOS. The workflow is depicted in Figure 8. It starts with a NEMO user of type *NEMO Partner* posting a request to add a new cluster into NEMO. This can be posted either through a direct call in the Intent-based API (assuming the user holds a valid authentication and authorisation token) or through the User Interface (UI) of NEMO's LCM. The request is dispatched to the Monetization and Consensus-based Accountability (MOCA) in order for the new resources to be considered in the accounting processes. Moreover, the request is dispatched to the Meta-Orchestrator, which actually includes the new resources in NEMO's managed clusters and makes the necessary configuration and initialisation processes, in order to deploy NEMO tools allowing for managed services in the new cluster, e.g. monitoring, user management, etc.

3.1.6.2 Workload Registration

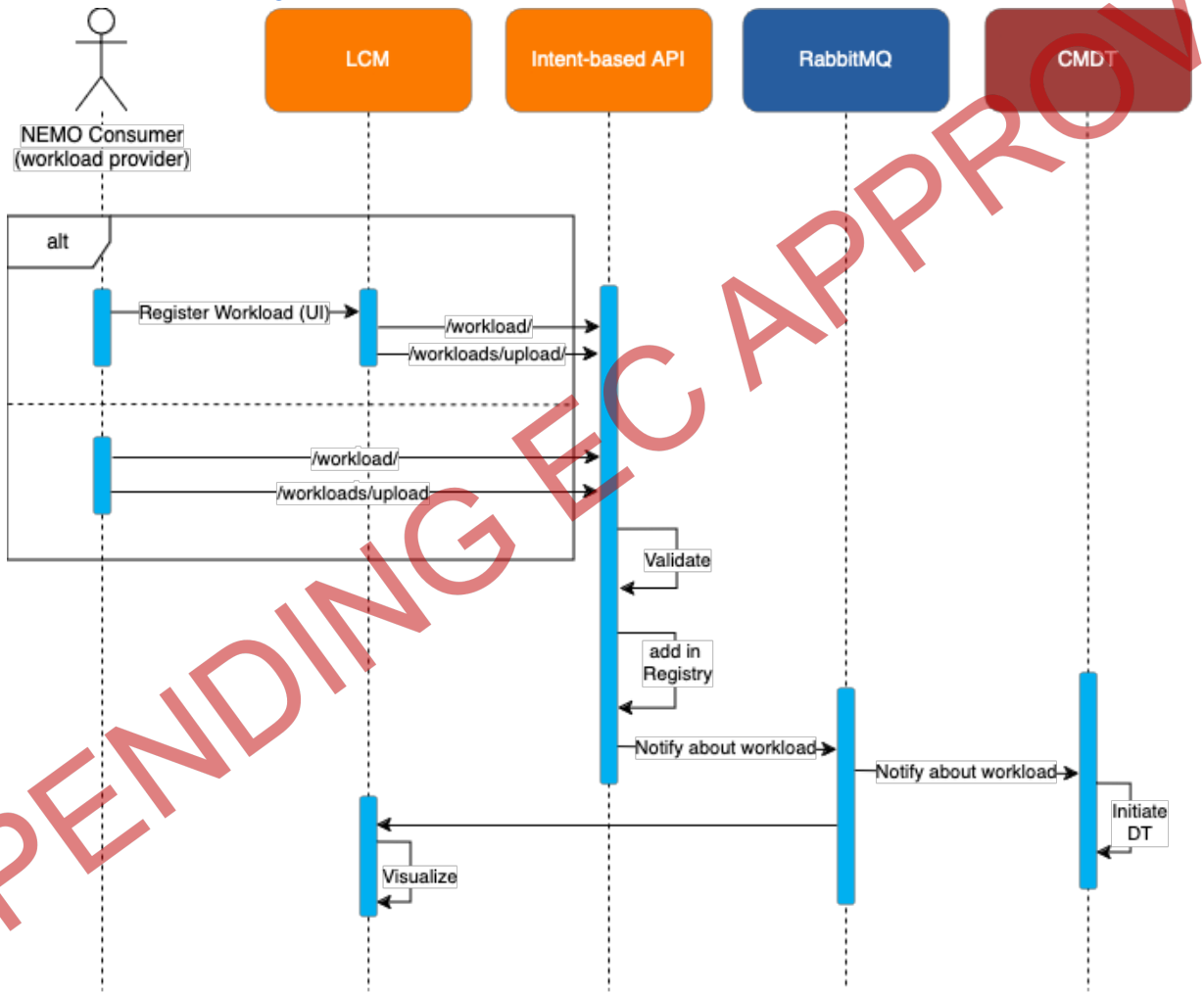


Figure 9: Process diagram for workload registration

Regarding the workload registration, NEMO implements the workflow depicted in Figure 9. First, NEMO consumers, desiring to deploy their application or service into NEMO, must create the workload documents for their workloads in the NEMO metaOS. This is realized by posting a relevant request to the NEMO Intent-based API directly or via the LCM user interface. The workload document includes information about the workloads per se (such as their name, version, schema and type), and the relevant intents for this workload. The intents include requirements for the execution of the workload, such as requirements for network, compute and storage resources, for secure execution (e.g., requirement to be executed through containers or unikernels), as well as energy efficiency requirements, such as executing the workload in green-powered servers, etc. Moreover, the workload document may include information for multi-cluster execution or execution in specific clusters.

Next, the NEMO Consumer has to make a request for upload, which involves uploading the workload helm charts. In this request, the NEMO consumer may declare resources of their workload that they desire to be exposed. In order to complete the registration into NEMO, the workload must successfully pass a set of validation checks, which verify that they are compliant with NEMO policies. These include compatibility checks between the workload versions and requirements and the NEMO clusters, workload assessment regarding discoverability of said resources, as well as security tests regarding the workload sources. Once the validation is successful, the workload descriptor is augmented with NEMO annotations and the workload is added to the NEMO Registry. Following this, the Cybersecure Microservices' Digital Twin (CMDT) is notified about the new workload.

3.1.6.3 Workload deployment

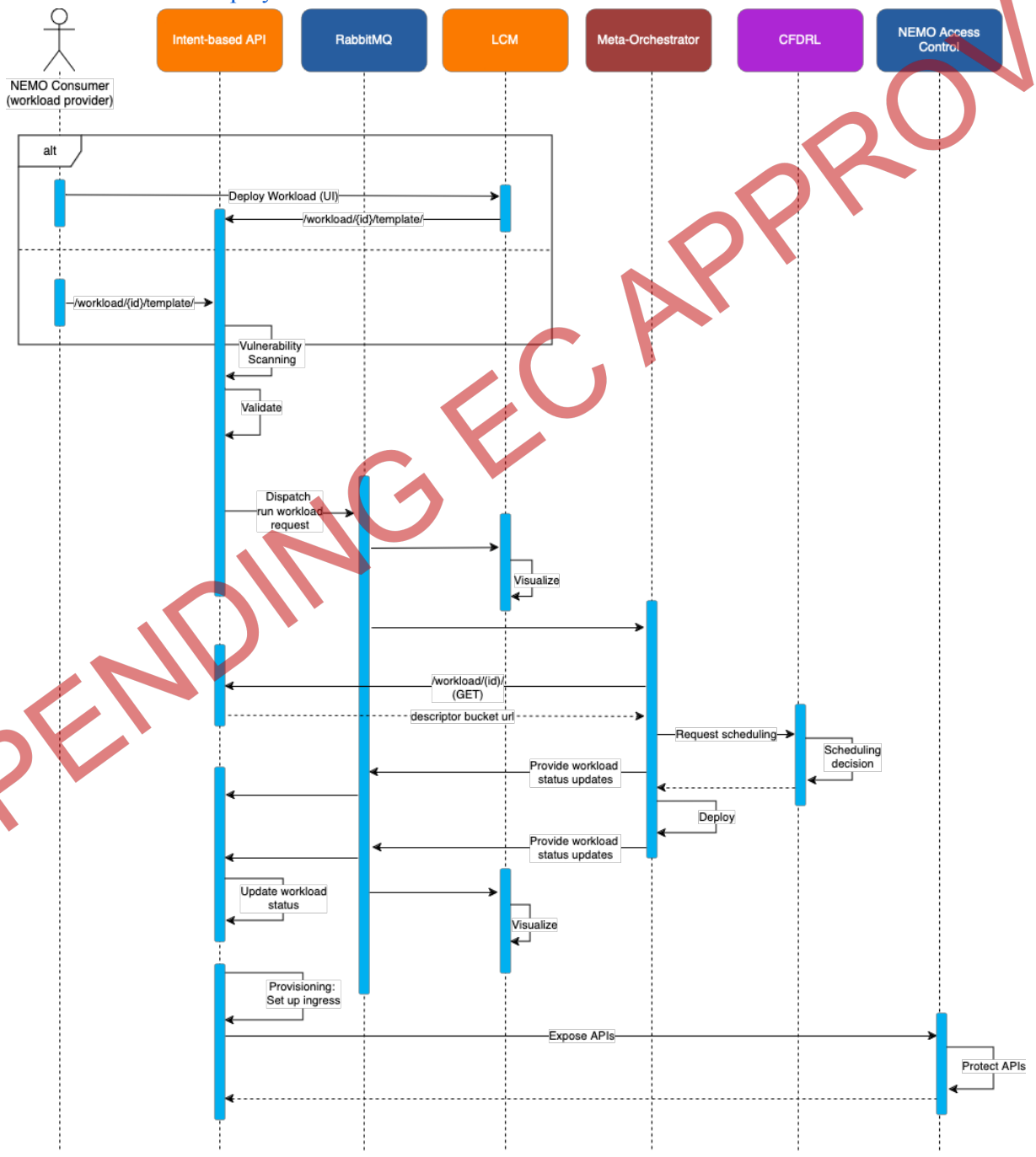


Figure 10: Process diagram for workload deployment

Once a given workload is registered in NEMO, the NEMO Consumer may submit a request for its deployment through the Intent-based API or via LCM UI. Before allowing the request to be forwarded for execution, validation and verification tests must have been successfully passed. These include compatibility, performance and security tests, as well. After successful verification and validation, the request for the workload deployment is communicated to the NEMO Meta-Orchestrator, which manages deployment considering both the specified intents and the NEMO resources' capabilities and policies. Upon successful deployment, the API caters for automated workload provisioning. This is realized by providing access to workload resources, following the defined Role-Based Access Control (RBAC) rules, ensuring that Ingress and Egress traffic are correctly set up, as well as workload Lifecyle Management is initiated for the deployed workload.

3.1.6.4 Workload intent's monitoring

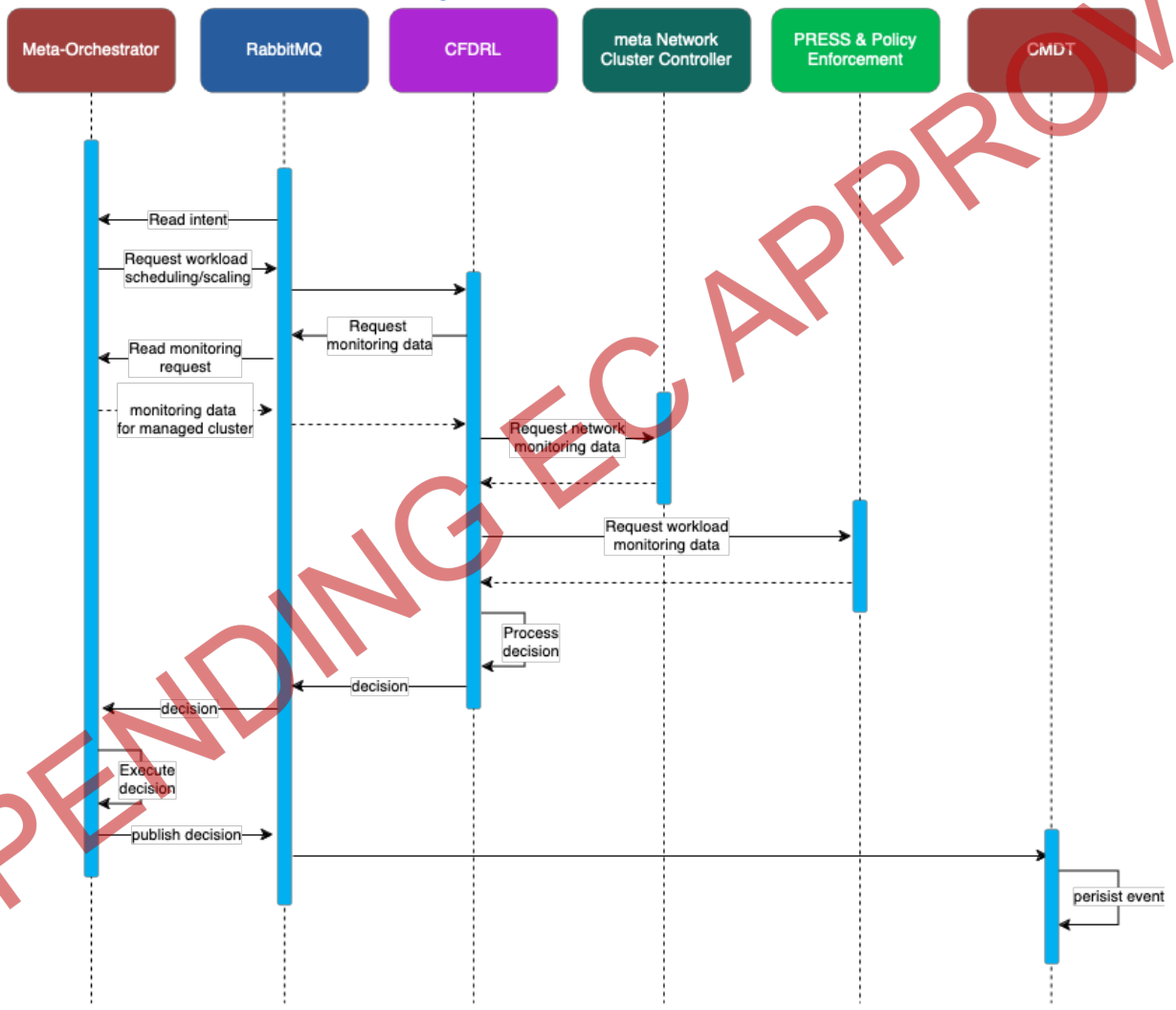


Figure 11: Process diagram for workload's intents' monitoring and enforcement

NEMO incorporates the concept of intents for the declarative description of requirements for workload execution and operation within the metaOS. Intent management processes are integrated by design in NEMO operations. Figure 11 presents the workflow for initiating, performing and exposing monitoring during workload execution. The process follows a running workload or a request for deployment. In any of these cases the Meta-Orchestrator requests the intents instantiated for the workload of interest. The request is communicated through RabbitMQ and is served by the Intent-based API (not shown in this figure). Once the Meta-Orchestrator holds the intents' information, it asks CFDRL for a recommendation regarding the placement or scaling of the workload. This request carries intents' information for the workload, as well as information of the capacity of NEMO's managed clusters. So

CFDRL is well aware of what kind of monitoring information it will request, in order to create an informed conclusion on workload's and resources' operational status. Hence, it posts the relevant requests for monitoring data to both mNCC for network monitoring data and PPEF for resource and workload monitoring data. These data feed CFDRL's relevant ML models and a recommendation on the workload action is inferred and communicated to the Meta-Orchestrator. The Meta-Orchestrator considers the recommendation just received and performs a workload action, which might be a deployment, migration or scaling. It then updates CMDT accordingly, which caters for tracking workload and resource information, in order to support data sovereignty and traceability with regards to the running workloads.

3.1.7 Development view

The development view in NEMO provides technical implementation details for the NEMO components. It is provided in the deliverables of WP2, WP3 and WP4, which detail the design options and development activities for the individual components.

3.1.8 Physical view

The physical view represents a topology map, guiding the deployment of the NEMO metaOS. Figure 12 presents an instantiation of NEMO in physical deployments, considering NEMO metaOS components running in the *management cluster* hosted in central cloud environment, while a set of managed clusters is integrated, including diverse resources and administrative domains, across all levels of the continuum. Table 3 suggests the deployment of NEMO components into six listed namespaces.

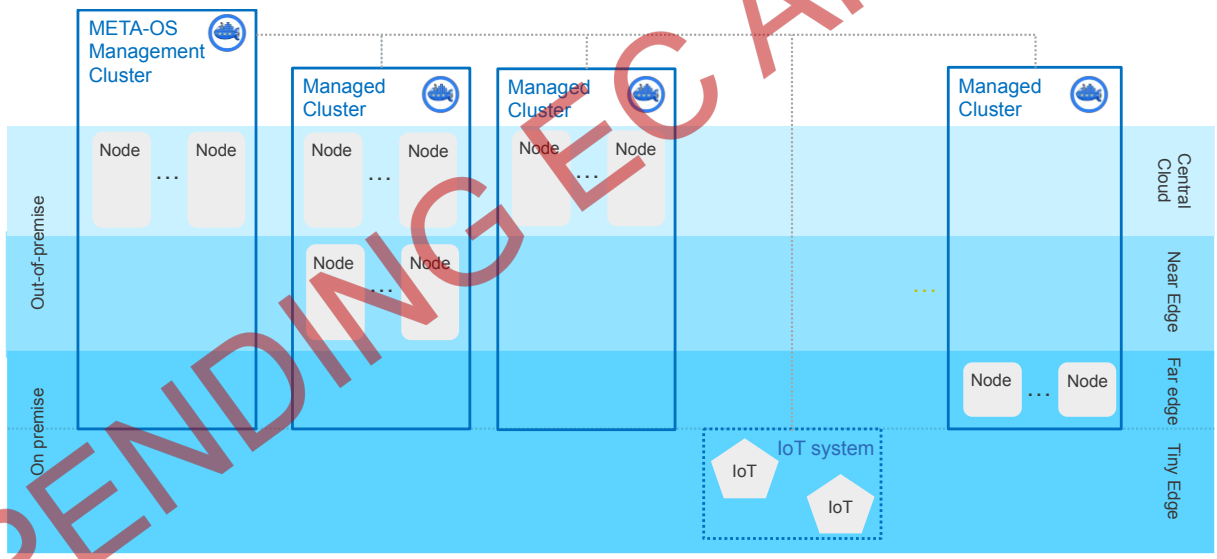


Figure 12: An instance of NEMO's physical view

Table 3: Namespaces in MetaOS management cluster

MetaOS namespace	Components
nemo-net	• Federated Meta-Network Cluster Controller • L2SM, 5G Connector
nemo-ai	• Cybersecure Federated Deep Reinforcement Learning • FREDY
nemo-kernel	• Meta-Orchestrator • Secure Execution Environment • Cybersecure Microservices Digital Twin

MetaOS namespace	Components
	• Intent-based Migration Controller
nemo-ppef	• PRESS & Policy Enforcement Framework
nemo-svc	• Plugin & Applications LCM • Intent-based API • Monetization and Consensus-based Accountability
nemo-sec	• Identity Management • Access Control • Secure Intercommunication (RabbitMQ)

3.2 MetaOS Architecture Coverage

Figure 13 presents essential functional elements for realizing the metaOS, placed at the different levels of the continuum. Since the metaOS environment is composed of highly dispersed devices, with significantly diversified power, network, communication and computing capabilities, differentiated instantiations of the functional elements are needed for constrained devices, edge and cloud resources.

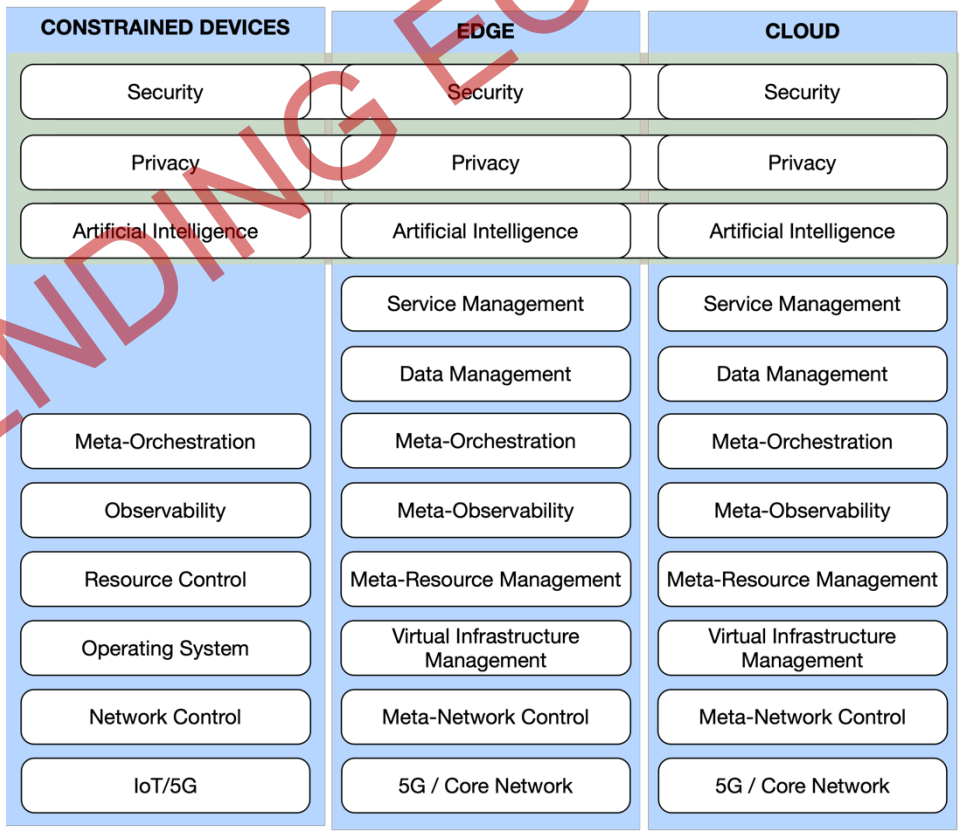


Figure 13: NEMO functional elements across the continuum: a functional stack vision

NEMO components addressing the functional elements identified per metaOS level, are presented in Table 4.

Table 4: NEMO components addressing MetaOS functionalities

MetaOS Functional Element	NEMO Component
Security	Identity Management Intercommunication Security NEMO Access Control Secure Execution Engine
Privacy	PRESS & Policy Enforcement Framework
Artificial Intelligence	Cybersecure Federated Deep Reinforcement Learning Federated Resilience Enhanced with Differential Privacy
Service Management	Intent-based API/SDK Lifecycle Manager MOCA
Data Management	Cybersecure Micro-services Digital Twin NEMO plugins
Meta-Orchestration	Meta-Orchestrator Intent-based Migration Controller
Meta-Observability	PRESS & Policy Enforcement Framework Cybersecure Micro-services Digital Twin Lifecycle Manager
Resource Management	Meta-Orchestrator MOCA
Network Management	meta-Network Cluster Controller 5G TSN API

The network infrastructure resources, namely IoT and 5G communications in the constrained devices and 5G or core network at the edge and cloud, are controlled via a meta-network control plane [15]. This meta-network cluster controller flexibly manages existing network management solutions, supporting secure micro-slicing and multi-path communication across the edge and cloud resources [16]. The physical resources in smart IoT devices are managed by embedded operating systems, while the edge and cloud resources are incorporated into Virtual Infrastructure Management (VIM) solutions, including cluster-based management (like K8s-based platforms) and virtual machine (VM) based platforms (e.g. cloud management solutions) [17].

Accordingly, resource onboarding and provisioning is addressed by meta-Resource Management at the cluster/VM level on top of such platforms in the edge and the cloud, as well as at cluster level for IoT devices. The metaOS addresses observability with intent-based policy monitoring and enforcement for cluster-based physical resources or IoT systems. Meta-orchestration federates the orchestration of computing workloads across the continuum, intelligently allocating workloads into available resources, through smart and flexible deployment, scaling and live migration.

Moreover, federated data management supports data collection, storage and sharing, ensuring data security and traceability. In NEMO, cybersecure microservices' digital twins address microservices' discovery and secure data access, integrating digital identities and Distributed Ledger Technologies (DLT). Service management provides workload lifecycle management, as well as accountability and monetization services to metaOS users, including the metaOS Provider, workload and resource providers, but also developers and technology providers.

Three cross-cutting elements are also identified, which are present across the different device levels, and every functional element. These include security compliance, privacy preservation and Artificial Intelligence.

Meta-Network Control is addressed via the Federated Meta-Network Cluster Controller, which is an automated, self-organizing entity that facilitates the dynamic creation and self-healing of fog IoT network clusters in the edge-cloud continuum connectivity. Moreover, intent-based network slices can be created through an open-source RESTful API, enabling TSN traffic in dedicated network slices.

Resource management is primarily supported by NEMO's Meta-Orchestrator, assisted by the Monetization and Consensus-based Accountability open-source components. The latter keeps track of the resources available in the metaOS for accountability purposes. The Meta-Orchestrator caters for resource provisioning and homogenized resource orchestration.

Meta-Observability [18] [19] is addressed via the PRESS and Policy Enforcement Framework by federating Service Level Objectives' (SLOs) monitoring taking place at individual clusters, while ensuring compliance to user-defined intents related to performance and energy efficiency requirements [20] at cluster and metaOS levels. Moreover, the Lifecycle Manager tracks events of broken objectives and additionally caters for secure audit logging.

Meta-Orchestration is core functionality of the NEMO metaOS. The open-source Meta-Orchestrator enables the decentralization and distribution of computing workflows across the IoT to Edge to Cloud Continuum. By acting as a central orchestrator, it manages the coordination and execution of complex distributed systems while addressing the challenges posed by their increasing complexity and heterogeneity [21]. The Meta-Orchestrator is complemented by the Intent-based Migration Controller (IMC), which facilitates seamless and efficient migration of computing workloads across distributed systems, encompassing IoT devices, edge computing infrastructure, and cloud environments.

The challenges regarding the Data Management of the running microservices [22] are addressed by NEMO's Cybersecure Micro-services Digital Twin (CMDT), which supports advanced data sovereignty.

CMDT supports microservice discovery in the metaOS, ensuring that every microservice and workload, from creation to retirement, is meticulously tracked and reported. In essence, CMDT acts as a reliable source of truth about the status and location of every microservice within its registry.

Service Management functionalities are provided by NEMO's open-source Intent-based API and SDK, as well as the LCM and MOCA components. The Intent-based API and SDK expose NEMO functionality through a set of resources of programmatic interfaces resources. Both API service description and implementation are foreseen, which facilitate external users in accessing the metaOS services, but also the NEMO system to limit access to its resources only to eligible entities and roles. Moreover, the NEMO LCM allows metaOS users to install and deploy registered applications, services, or plugins to the metaOS automatically and transparently to the user. Based on users' intents, the NEMO LCM performs automated continuous delivery operations, security controls and event-based responses, following Infrastructure as Code (IaC) principles. In addition, MOCA realizes innovative smart contracts' based business models for monetizing the metaOS resource usage, implementing a consensus-based distributed architecture for sharing networking, computing, and storage resources from various end-users and (competing) telecom and cloud providers.

Regarding to the crosscutting functions, AI is vertically present in the metaOS architecture. The Cybersecure Federated Deep Reinforcement Learning (CFDRL) component provides capacity of learning decision-making models in a collaborative and distributed way between communicating nodes/entities. The learning procedure combines two complementary learning paradigms: Federated Learning (FL) and Reinforcement Learning (RL). In order to address privacy preservation challenges in knowledge aggregation and transfer, and, subsequently, in CF-DRL, NEMO introduces FREDY (Federated Resilience Enhanced with Differential Privacy) [23] which integrates Flower with Private Aggregation of Teacher Ensembles (PATE) [24] to bolster privacy features.

Security in the metaOS is built on the concept of ZeroTrust. NEMO Access Control (NAC) allows the implementation of a comprehensive approach to applying flexible, easily configurable, granular

privileged access to NEMO resources by either internal components or, beyond the perimeter, to external entities. NAC provides a security substrate to NEMO resources, enforcing that any attempted connection is brokered through a common API gateway. Then, access control is applied based on a set of modular criteria, which may include identity management, catering for Authentication, Authorization, and Accounting (AAA), but also traffic flow controls and universal controls for specific IPs through whitelist and blacklist rules.

Moreover, the intercommunication security module of NEMO is based on a message broker, enabling communication and synchronization among distributed systems and applications. It acts as an intermediary, facilitating secure message exchange while offering essential capabilities like message routing, queuing, and transformation. In addition, NEMO's Secure Execution Engine (SEE) enhances Kubernetes via its a Unikernel runtime and its NEMO migration extension for K8s, which is necessary to enable the migration of microservices across nodes in the IoT-Edge-Cloud continuum.

Furthermore, security features are integrated within other components, too. The LCM provides secure audit logging, and the API performs IaC vulnerability scanning. In addition, mNCC enables network slice creation for secure microservice isolation. Secure AI operations are addressed via privacy preserving Federated Learning, as well as FL attack detection and mitigation components.

3.3 MetaOS Architecture Extensions

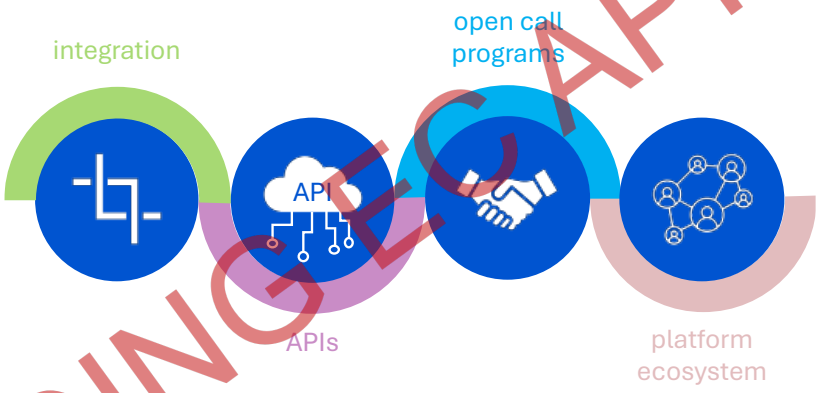


Figure 14: NEMO platform ecosystem building

The idea of the meta-Operating System (metaOS) refers to effective integration of highly diversified software and hardware resources, enabling homogenized flexible orchestration of diverse workloads across heterogeneous and dispersed devices. In NEMO, the architecture is extensible, allowing further functionalities to be added as NEMO plugins. With the NEMO Kernel in the role of the core system, plugins are meant to provide additional features as plugins to the core, providing extensibility, flexibility, and isolation of application or custom metaOS logic. Plugins may refer to horizontal services or domain-independent services that aim to provide some basic and common functions that extend the NEMO capabilities.

NEMO strategy for ecosystem building is depicted in Figure 14, identifying four main steps towards realizing NEMO as a SaaS, PaaS and IaaS platform, on which third parties may build upon. These include *integration*, *APIs*, *open call programs* and finally the *platform ecosystem*.

Integration is the simplest way of onboarding third parties on NEMO and may refer to integration of either workloads, resources or data. NEMO provides integration points for potential application, service and AI providers, wishing to deploy their products on NEMO metaOS. Similarly, hardware providers, which may host metaOS workloads, can be easily integrated as NEMO resources, spanning IoT, edge and cloud devices. Moreover, data providers have the possibility to integrate their historical or real-time data streams into the metaOS. Integration at this level is user-friendly, exposing graphical and programmatic capabilities with minimal deviation from common development practices. In addition, NEMO suggests and implements business models for monetising workloads, resources and data.

Moreover, it is flexible enough to support other business models, which would be implemented as smart contracts.

Exposed *APIs* are foundational features for ecosystem building. NEMO's Intent-based API aim to expose NEMO lower-level functionality to the outside world in an easily accessible format, minimizing the effort needed on their side to adapt applications, services and plugins to NEMO-capable ones, but also introducing minimal distraction compared to common practice for proficient cluster users. The API will automatically discover and expose resources of registered and deployed workloads in NEMO, enforcing privileged control to their access. Moreover, the NEMO API supports the registration and deployment of workloads into the NEMO resources.

NEMO has already realized extensions through additions developed by the 1st Open Call projects. Six winning proposals have been identified, specifically MetaFOX, by Vodéna, Serbia working on Machine Learning and AI technologies; CorMOS, by Business & IoT Integrated Solutions, Cyprus focusing on cross-domain orchestration of MetaOS edge resources; ARGO, by Intellia ICT, Greece, investigating porting AR/XR technologies in the metaOS; Eros4NRG, by Martel Innovate BV, Netherlands offering ZeroTrust IoT Analytics with focus on Smart Energy applications; GENESYS, by SWHARD srl, Italy providing an edge gateway for the NEMO MetaOS; and MARINEMO, by BEAM Innovation, Romania working on an efficient resource utilization and maritime network slicing plugin for the NEMO MetaOS.

3.3.1 MetaOS and Data Spaces

Data Spaces might be relevant for the metaOS in 2 directions. First, the metaOS's control flow might be integrated into a dedicated Data Space, e.g. metaOS Data Space, allowing controlled exchange of relevant data and logs across different metaOS implementations, but also among metaOS stakeholders, such as the metaOS Provider, the metaOS Consumer (Application Provider) and the metaOS Partner (Resource Provider). On the other hand, the metaOS may integrate with Industrial Data Spaces (IDS) for different application verticals, serving as the venue for secure data sharing and monetization. In that respect, the metaOS may host the services that implement the roles of the Service Provider, the Broker Service Provider, the Clearing House and the App Store Provider, as defined in the IDSA Reference Architecture Model [25]. Figure 15 presents the metaOS positioning among the Data Space roles and interactions. At the same time the metaOS will ensure that these services are executed across metaOS resources with assured performance, security requirements, energy efficiency targets, etc.

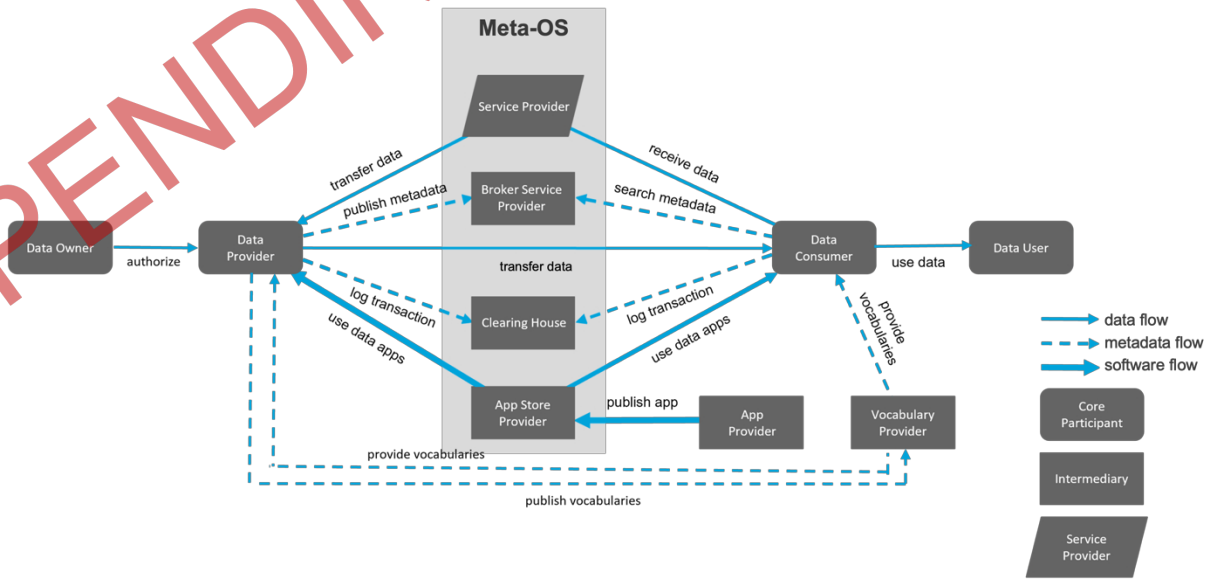


Figure 15: MetaOS positioning across IDS roles and interactions

In order to integrate IDS, additional metaOS functionalities must be in place, beyond those of the core metaOS components. In NEMO, the architecture is extensible, allowing further functionalities to be added as NEMO plugins. With the NEMO Kernel in the role of the core system, plugins are meant to provide additional features as plugins to the core, providing extensibility, flexibility, and isolation of

application or custom metaOS logic. Plugins may refer to horizontal services or domain-independent services that aim to provide some basic and common functions extending NEMO capabilities.

3.3.2 Case study: FIWARE based Data Space Connector for NEMO

The main objective of the IDS Connector is to ensure data privacy and sovereignty, as well as interoperability among the Data Space stakeholders, especially the Data Providers and Data Consumers. A Data Space Connector relevant for managing application data exchanges within the metaOS can be implemented based on the architecture depicted in Figure 16 [26]. The proposed IDS connector comprises the FIWARE Orion Context Broker [27], and as well as the relevant IoT agent [28] and system adapters for connecting with devices and systems, respectively, acting as data sources. Moreover, it includes an Access Control component, acting as Policy Enforcement Point and allowing to keep control of the data.

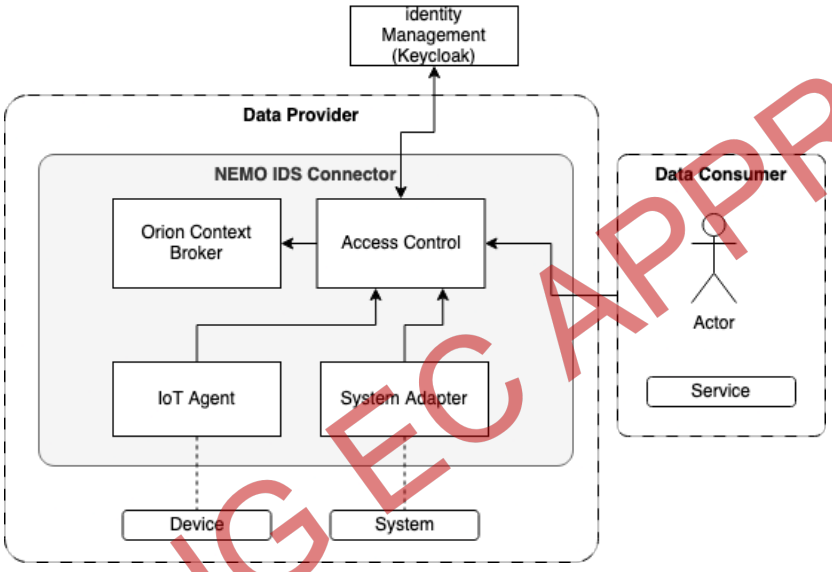


Figure 16: Indicative IDS Connector for integrating NEMO with Data Spaces

FIWARE [29] offers several open API (usually open source as well) reusable components, called Generic Enablers (GEs), that can be used in a variety of contexts and applications, including IoT. Indeed, several IoT platform solutions have been built based on FIWARE [30] [31]. In general, the architectural approach aims to allow devices to connect to an IoT Agent, which abstracts the complexity of the networking protocols and translates the relevant information into data models and protocols compliant with the OMA Next Generation Service Interfaces (NGSI) 9/10 specifications [32]. Moreover, it aims to push the device-provided information to a context broker supporting the OMA NGSI 9/10 standard. From this context broker, external systems such as Complex Event Processing Engines or third-party applications, may get updates on the devices' identities and status.

Using the NEMO Access Control component for the relevant component of the IDS Connector allows the implementation of a comprehensive approach to applying flexible, easily configurable, granular privileged access to metaOS resources -and thus data- by either internal components or, beyond the perimeter, to external entities. It provides a security substrate to NEMO resources, enforcing that any attempted connection is brokered through a common API gateway. The Access Control protects internet-exposed endpoints from unprivileged data breaches. It is appropriate for the hybrid and highly diverse environment within the metaOS.

Every access request can be evaluated against a set of defined access control criteria. This allows the application of chaining access control criteria, which may differentiate among endpoints or even user roles. In any case, AAA controls are applied, relying on Keycloak as the identity management solution. Additional criteria might include traffic flow controls and universal controls for specific IPs through whitelist and blacklist rules. Once the evaluation check is successful and defined access control criteria are respected for both the endpoint in question and the entity requesting access (user, third-party or

NEMO service), then the request is routed to the relevant Orion Context Broker endpoint. This leads to the request being processed and the relevant response to be communicated to the requester in a synchronous manner. In the opposite case, in which at least one control criterion fails, access is not granted, and the requester is provided with a relevant response text and code, indicating that access to the requested resource is forbidden. In this way, this IDS Connector addresses functionalities of the Connector Core Services, identified by IDSA [33], namely *Authentication Service*, *Policy Engine* and *Data Management*.

3.4 Alignment to EUCEI Reference Architecture

NEMO has been one the research projects funded by the EC to develop meta-operating system as a future platform for the edge, funded under the topic HORIZON-CL4-2021-DATA-01-05 [34]. NEMO and other research and innovation actions funded under this or similar topics, approach the challenge of metaOS building from different perspectives, addressing various aspects of the continuum. The projects' efforts towards developing platform and technologies around the IoT, edge and cloud are coordinated within the EUCEI community [35]. The EUCEI project members are collaborating in the metaOS building around diverse concepts, including strategic liaisons, open-source engagement, architecture, ecosystem engagement, market & sectors and communications.

On the architectural side of metaOS development, a common Reference Architecture has been developed for the cloud-edge-IoT continuum, incorporating the efforts of EUCEI members. In the following subsections, the NEMO architecture is mapped into different views of the EUCEI architecture.

3.4.1 Functional View

EUCEI architecture definition is based on eight *building blocks* [36], corresponding to NEMO's *Functional View*. EUCEI's *building blocks* are mapped to NEMO's *functional blocks*, as depicted in Figure 17, where the circles with the ticks on each building block indicate by their colour the NEMO functional block which delivers similar functionality.

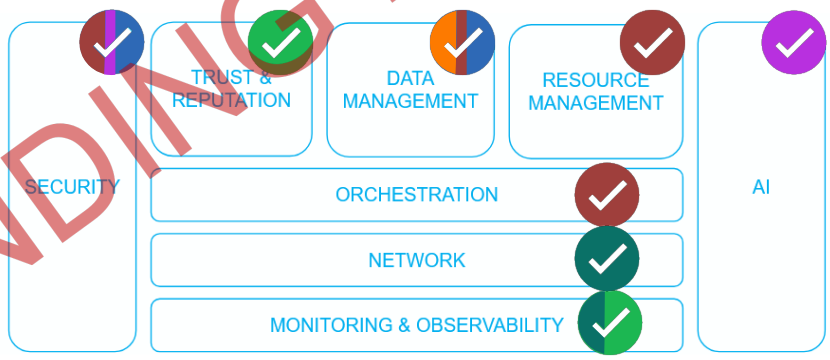


Figure 17: NEMO Alignment to EUCEI RA functional view [36]

EUCEI identifies the minimum set of functionalities included per building block.

The *Monitoring & Observability* services in EUCEI cover automated monitoring, system health, SLA management, compliance and observability. In NEMO such capabilities are part of the NEMO PRESS & Policy Enforcement Framework (PPEF), grouped under the umbrella of intent management and enforcement.

Network management urges for essential abstraction and programmability of network infrastructure. This is addressed by the *Network* EUCEI block, while NEMO similarly identifies a dedicated horizontal block, namely Infrastructure Management. Network functions such as slicing, tunneling, mobility, Time Sensitive Networking (TSN) in 5G and beyond are envisaged in both EUCEI and NEMO.

In addition, core functionalities of metaOS are envisaged for the orchestration services. Thus, *orchestration* in both EUCEI and NEMO are at the centre of the architectures and undertake orchestration of both services (workloads) and resources (host infrastructure) to abstract underlying heterogeneity and provide homogeneous management of both, including workload deployment,

migration and scaling, load balancing, lifecycle management, service brokerage, as well as intelligence support in workload placement and migration. NEMO Kernel functional block addresses these functionalities, delegating the intelligence part to the CFDRL vertical layer.

The *Resource Management* building block in EUCEI deals with federation, ad-hoc clustering, discovery, registry and communication services. Such capabilities in NEMO are delivered primarily by the NEMO kernel as part of the meta-orchestration responsibilities, assisted by the microservices' digital twinning. Moreover, registry services are provided by NEMO's Service Management layer.

The *Trust & Reputation* building block includes trust algorithms, reputation mechanisms, traceability and accountability. The Service Management functional block of NEMO addresses traceability and accountability, as well as data sovereignty in relevant transactions, while trust and reputation could be part of NEMO PPEF.

Moreover, the *Data Management* building block in EUCEI reference architecture embraces operations around data, such as processing, curation, interoperability, semantic annotation, thinning patterns definition, discovery, consistency, availability and storage. Thanks to NEMO flexibility, data management is addressed through NEMO extensions in the Service Management Layer.

For *Security*, the functionalities identified include security, privacy, anomaly detection, smart contracts, authorization. In NEMO, this functionality is mostly addressed by the Cybersecurity & Unified/Federated Access Control (blue box in the functional view), which is -similarly- designed to address security needs as a vertical layer in the architecture. As security mechanisms and tools are designed specifically for workload execution and AI modules, CFDRL and NEMO Kernel layers are attributed part of this functionality in the figure.

The second vertical building block in EUCEI refers to *Artificial Intelligence* (AI), addressing frugal AI, tiny AI, distributed AI and Federated Learning. NEMO identifies a dedicated vertical layer for AI, too, referred as *NEMO Cybersecure Federated Deep Reinforcement Learning*. This NEMO block, addresses federated learning, reinforcement learning, as well as these too integrated, gossip learning, ML operations, including model storage, sharing and serving, as well as cybersecurity in ML.

3.4.2 Compositional View

Each of the EUCEI RA building blocks is further analysed in development needs through the RA's *Compositional view* [37], which provides information about metaOS capabilities, at a finer level of detail, which is useful for development teams. Similar information is provided in NEMO through the development architectural view [1]. In the following, a further mapping of individual components is described among the two approaches.

3.4.2.1 Security

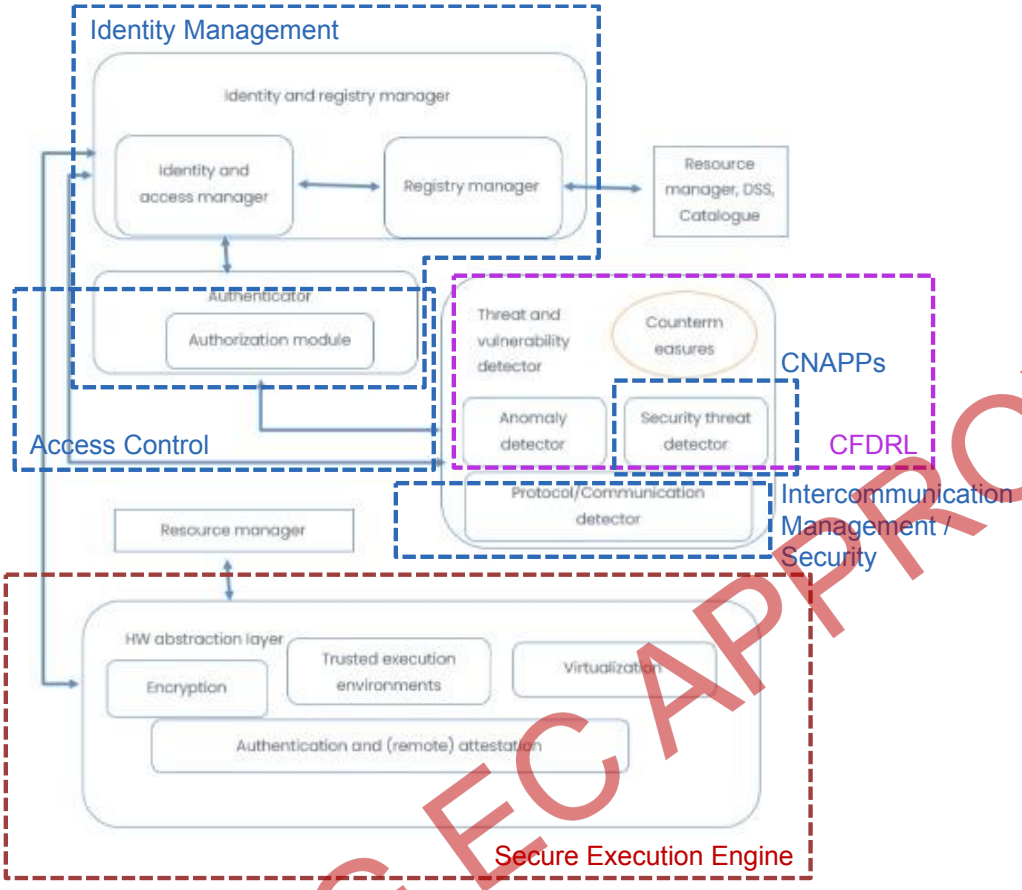


Figure 18: Security and privacy components of EUCEI RA mapped to NEMO

The components addressing security and privacy in EUCEI RA include (Figure 18):

- Identity and registry manager: NEMO identifies *Identity Management* for similar support.
- Authenticator: The Identity Management and the *Access Control* components address authentication and authorization services in NEMO.
- Threat and vulnerability detector: In NEMO, vulnerabilities, threats, attacks and mitigation are considered in different levels, mostly as part of the defined *CNAPPs* [38], while the relevant functionality whose design is based on ML tools and services is incorporated within *CFDRL*. Specifically, NEMO considers Privacy Preserving Federated Learning and delivers the FREDY (Federated Resilience Enhanced with Differential Privacy) framework as implementation of it. Moreover, attack detection in training and inference data streams, as well as relevant countermeasures complement combined FL/RL modules within NEMO's CFDRL [39].
- HW abstraction layer (for security): Trusted execution environments and virtualization are present in NEMO as the *Secure Execution Engine*.

3.4.2.2 Trust and reputation

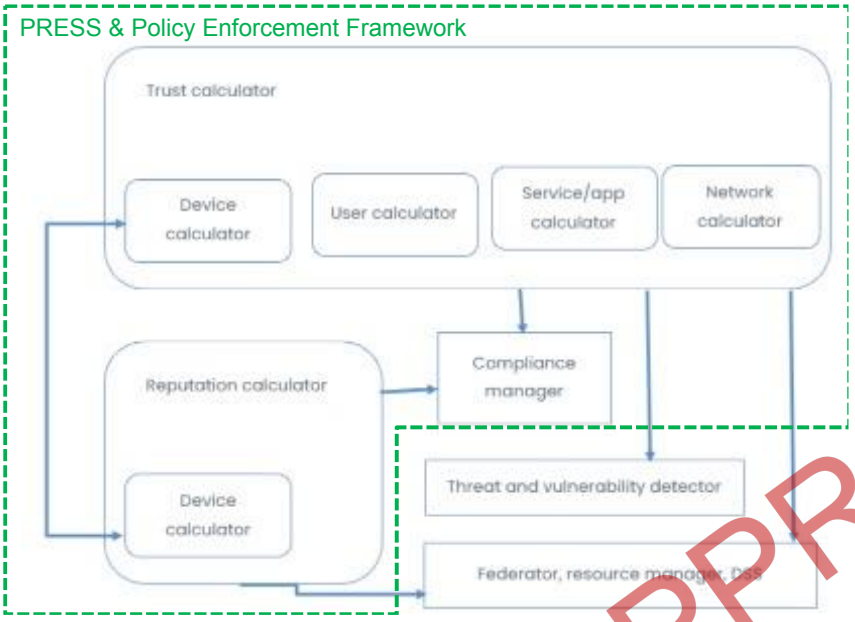


Figure 19: Trust and reputation components of EUCEI RA mapped to NEMO

The components addressing trust and reputation in EUCEI RA include (Figure 19):

- Trust calculator: NEMO does not foresee a dedicated component for this functionality, but it foresees PRESS monitoring and compliance within the PPEF.
- Reputation calculator: NEMO does not foresee a dedicated component for this functionality, but it can be supported as additional metric calculator within PPEF.

3.4.2.3 Data management

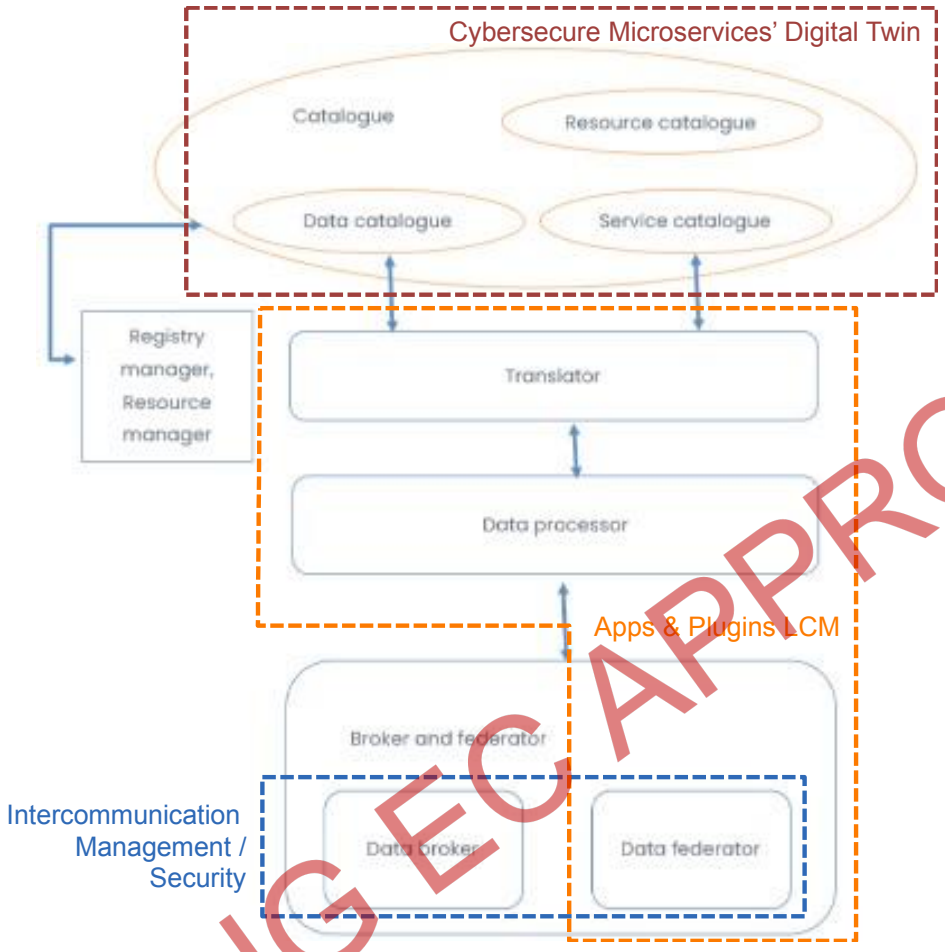


Figure 20: Data management components of EUCEI RA mapped to NEMO

The components addressing data management in EUCEI RA include (Figure 20):

- Data catalogue: Discovery and catalogue within NEMO are addressed by the CMDT component.
- Data broker and federator: NEMO's Intercommunication Management / Security module is foreseen as a message broker which serves for message exchange, providing features such as message routing, queuing, and transformation [38]. The Apps & Plugins Lifecycle Manager addresses data federation and may integrate a set of data connectors (e.g. DBs, Data Spaces, blockchain networks) as NEMO extensions.
- The Translator (the lowest level to connect DBs) ensuring interoperability, uses data catalogue for information on data models. NEMO's Apps & Plugins Lifecycle Manager may integrate such components as plugins.
- Data processor (also providing information for scheduling): NEMO's Apps & Plugins Lifecycle Manager may integrate such components as plugins.

3.4.2.4 Resource Management

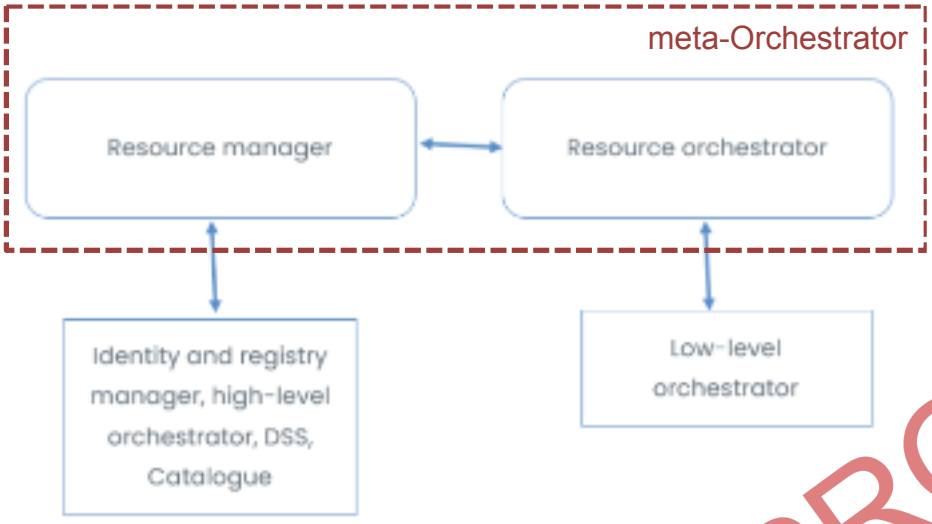


Figure 21: Resource management components of EUCEI RA mapped to NEMO

The components addressing resource management in EUCEI RA include (Figure 21):

- Management and scheduling module: In NEMO, the meta-Orchestrator is responsible for managing computational resources across the IoT, edge and cloud continuum.
- Allocation module: The NEMO meta-Orchestrator covers resource allocation for workload execution, consulting CFDRL for intelligent decision making.
- Discovery module: The NEMO meta-Orchestrator with the help of CFDRL ML models for workload scheduling is responsible for resource selection per required activity.

3.4.2.5 Orchestration

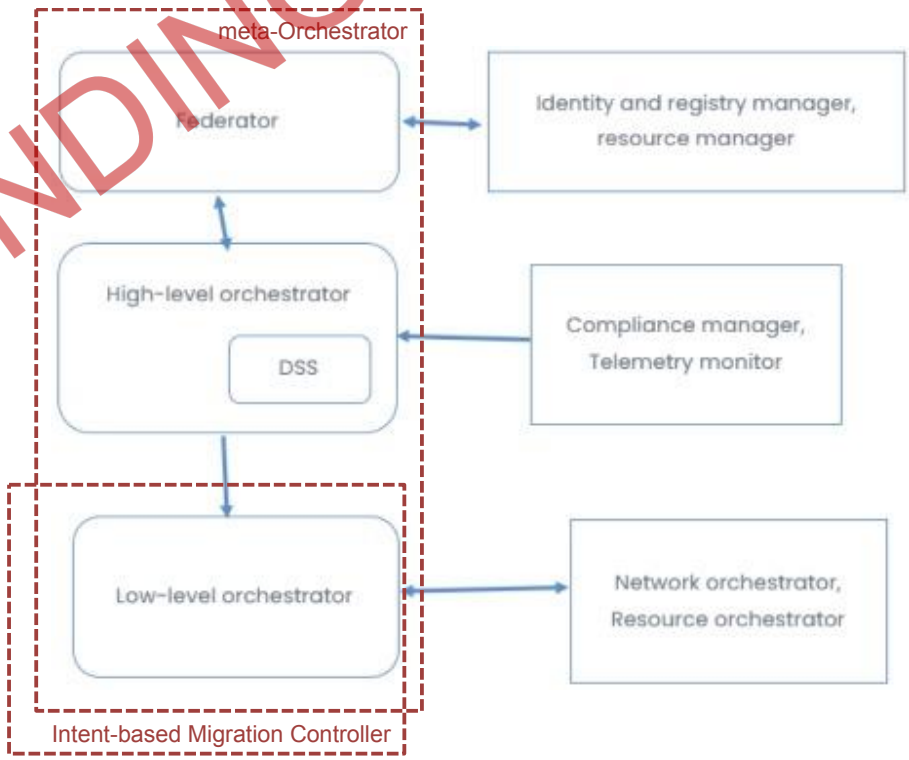


Figure 22: Orchestration components of EUCEI RA mapped to NEMO

The components addressing orchestration in EUCEI RA include (Figure 22):

- Federation: This is dealt by the Meta-Orchestrator in NEMO, offering a control plane on top of all integrated resources into the metaOS.
- Service description: This is addressed by the Intent-based API, which caters for workload upload, validation and registration into the metaOS registry.
- High-Level Orchestration: The NEMO Meta-Orchestrator is responsible for the dynamic orchestration of workloads in available resources, ensuring workload requirements for their execution. In other words, the NEMO Meta-Orchestrator manages workload deployment, migration and scaling, in case there are relevant recommendations from the CFDRL side, the latter considering monitored performance as per defined intents.
- Low-Level Orchestration: The NEMO Meta-Orchestrator addresses general control actions involving both workload and resources, such as deployment and scaling. In addition, workload migration tasks, specifically, fall within the scope of NEMO's Intent-based Migration Controller.

3.4.2.6 Network

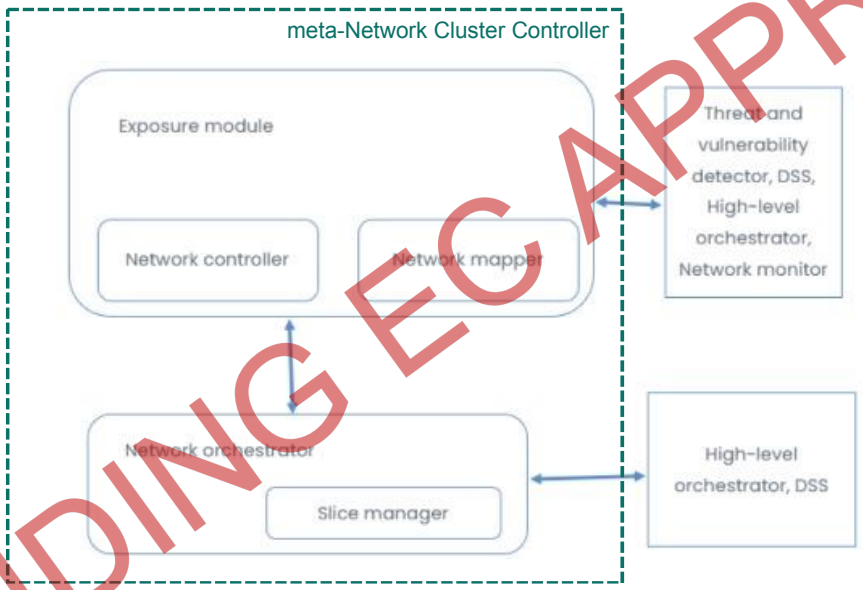


Figure 23: Network components of EUCEI RA mapped to NEMO

The component delivering network capabilities at metaOS level in the EUCEI RA include (Figure 23):

- Exposure module: Network probing, assessment and exposure of network characteristics are incorporated into NEMO meta-Network Cluster Controller (mNCC).
- Network Orchestration: NEMO's mNCC delivers such orchestration, overseeing network connectivity management and ensuring multi-cluster connectivity and slice management.

Moreover, mNCC incorporates technology connectivity adaptors, which convert data between different network protocols, enabling devices and systems using disparate protocols to communicate effectively. These include:

- L2S-M, which enables the dynamic creation and management of isolated virtual networks within metaOS clusters.
- 5G adaptor, which supports deterministic communications through TSN bridges with 5G LAN solutions. The 5G adaptor allows for applying, modifying and deleting data flows, while supporting monitoring and analytics capabilities.
- SDN-based connectivity adaptor, supporting network management and programmability.

3.4.2.7 Monitoring and Observability

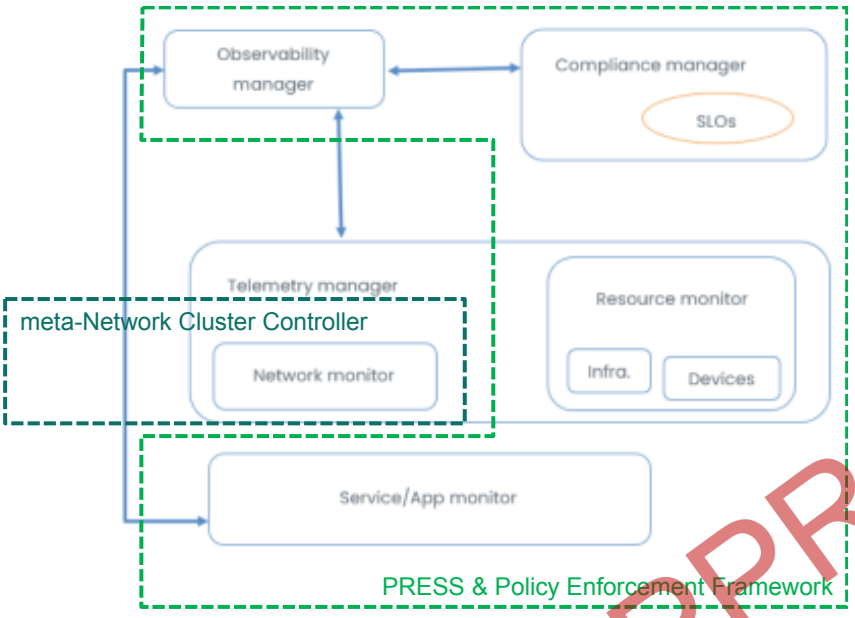


Figure 24: Monitoring& observability components of EUCEI RA mapped toNEMO

The components envisaged in EUCEI RA for delivering monitoring and observability include (Figure 24):

- Service/app monitor: The PPEF component of NEMO delivers relevant workload monitoring capabilities and exposes them at metaOS level.
- Resource monitor: Similarly, for resource monitoring NEMO PPEF caters for both probing and metrics' exposure, related to computing resources (e.g. servers or other physical devices)
- Network monitor: mNCC deals with network probing and exposure to PPEF.
- Observability Manager: PPEF manages the monitoring information about workload defined requirements in the form of intents.
- Compliance Manager: Based on the monitoring feedback, as received from workload-level aggregated observability, PPEF concludes on whether the defined intents per workload are fulfilled or not, while they are running.

3.4.2.8 Artificial Intelligence

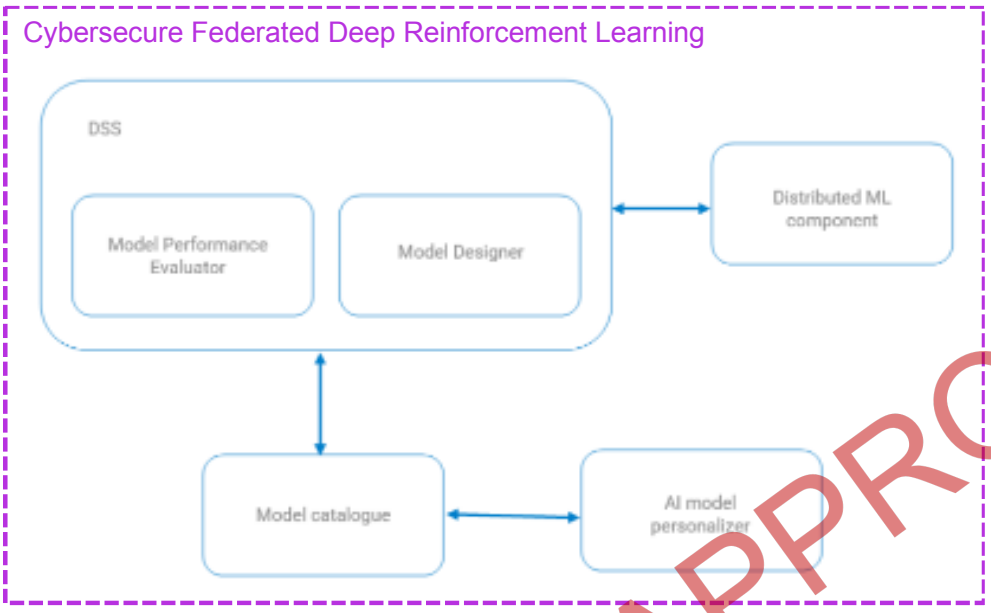


Figure 25: Artificial Intelligence components in EUCEI RA mapped to NEMO

The AI components envisaged in EUCEI RA include:

- Decision Support System: This is also foreseen as functionality of the NEMO CFDRL, incorporating ML models to support the Meta-Orchestrator in making workload scheduling (deployment, migration) and scaling decisions.
- Model catalogue (optional): A model storage component is also part of CFDRL.
- AI model personalizer (optional): CFDRL is seen as a family of ML operations, that support Federated Learning, Reinforcement Learning, model sharing and serving. As such, AI model personalisation is also addressed by CFDRL.
- Distributed ML component (optional): This is also addressed by Federated Learning component of CFDRL.

3.4.2.9 Additional NEMO functionalities

NEMO envisages an accounting mechanism on top of the designed MetaOS. It is considered critical in providing visibility, control, and cost management across multi-cluster environments. By tracking resource usage and managing costs, it can assist in enforcing policies and generating insights in terms of optimised resource utilization and expenses' control. This leads to better resource allocation, enhanced operational efficiency, and improved financial management in cloud-native environments.

Such functionality is part of the Service Management layer and specifically provided by the NEMO Monetization and Consensus-based Accountability (MOCA) component, where Distributed Ledger Technology (DLT) technology verifies data integrity.

MOCA aims to support monetization and accountability for both the applications and plugins running on NEMO. It implements a consensus-based distributed architecture for sharing networking, computing and workload resources across different stakeholders and end-users.

3.5 Alignment for Use Cases Architectural patterns

Inspired from architectural patterns as reusable solutions in software design, architectural patterns are suggested for the categorization of use cases applied across the continuum, by EUCEI. The suggested architectural patterns have been investigated for use cases in the Agricultural, Constructions, Energy, Healthcare, Manufacturing and Transport sectors and refer to:

- L1: Location tracking for people, animals and devices.
- L2: Visual inspection based on images and videos analysed for surveillance and inspection for people and devices (things, products, etc.)
- L3: Condition monitoring based on telemetry data analysed and referring to people, fixed or mobile devices, risk/hazard, operations, etc.
- L4: Predictive maintenance, based on analytics over telemetry data which support reporting and diagnostics for maintenance of fixed and mobile devices.
- L5: Command/Control, which refers to actuation capabilities enabled over systems and devices.
- L6: Process Management / Autonomous Operations, referring to systems with high AI penetration and automation, potentially including autonomous robot systems, vehicles or even buildings with automated management operations.

NEMO covers use cases in the Agriculture, Energy, Manufacturing, Transport and Media sectors. Table 5 lists the use cases considered in the NEMO Living Lab trials, categorized per EUCEI use cases architectural patterns [40].

Table 5: NEMO use cases mapped in EUCEI L1-L6 architectural patterns

Sector	L1: Location tracking	L2: Visual inspection	L3: Condition monitoring	L4: Predictive maintenance	L5: Command/ Control	L6: Process Management / Autonomous Operations
Agriculture		Product inspection	Hazard, Field		Yes	AGV. Process management
Energy			Hazard, Fixed equipment, operations	Fixed equipment, operational diagnostics & reporting		Process management
Manufacturing	Employee, movable assets	Product inspection	Employee, hazard		Yes	AGV, process management
Transport		Inspection	Electric Vehicles	Operational reporting		Process management
Media	Runner	Surveillance, Inspection	Telemetry, remote monitoring	AI analysis & recommenda-tions	Yes	AR/XR interactions, process management

As use case categorization in application verticals within the NEMO project is as defined in section 2, Figure 26 indicates the NEMO Living Labs, which foresee validation of use cases for each of EUCEI use case architectural pattern.

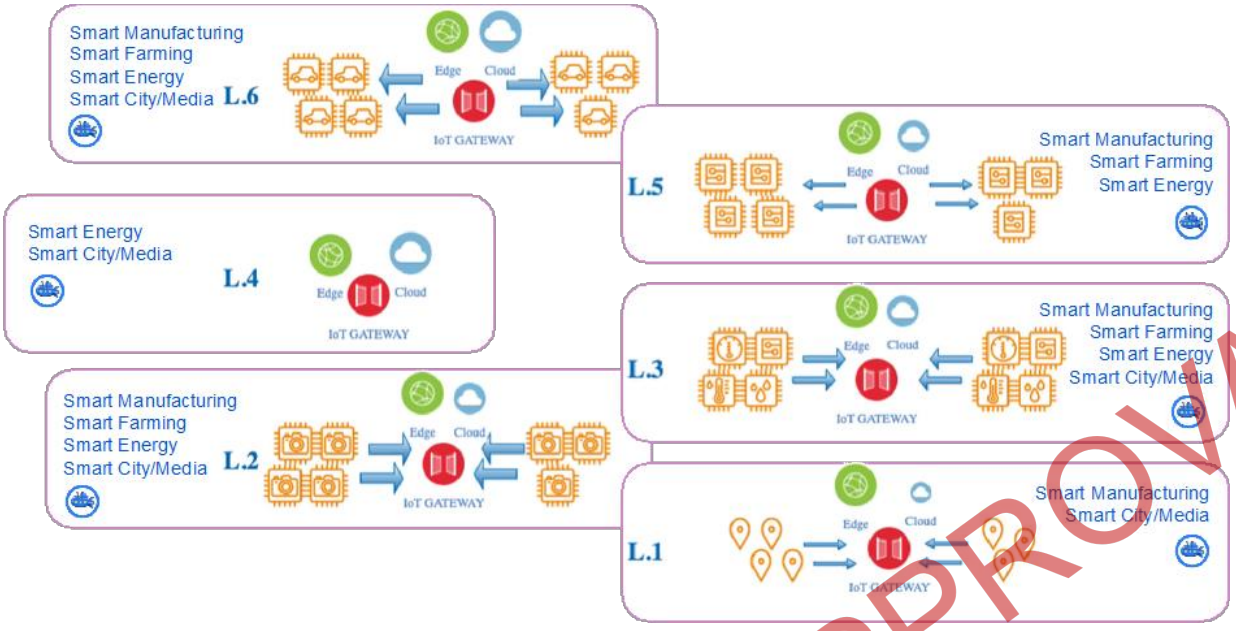


Figure 26: NEMO Living Labs validating EUCEI use cases architectural patterns

4 V&V methodology - updates

NEMO provides a holistic approach for the evaluation of each new workload before it is deployed in the continuum. As several third-party microservices are deployed to heterogeneous infrastructures (i.e., central cloud, edge, IoT devices) using different technologies (i.e., VMs, containers, k8s, Swift, etc.), NEMO must ensure that the specific workload fulfils the requirements and Key Performance Indicators (KPIs) and that it is compatible with the innovative features that the NEMO platform offers (i.e., resource scaling, high availability, full-stack automated operations, etc.). For this reason, a set of test types is adopted to enhance services' security, reliability, and performance. NEMO provides an intrinsic component named Validation and Verification environment for test execution; however, some tests can be executed in other components (e.g., Syntax testing runs on the Intent API component). As mentioned in D1.2, the architectural approach is modular and integrated with NEMO's DevOps cycle. In the following sections, we present the test types that NEMO will offer and the appropriate testing tools and frameworks that will be supported.

4.1 Testing Types

4.1.1 Acceptance testing

Acceptance testing is a crucial phase in the testing process that ensures the developed services fulfil the specified user expectations and are compatible with the NEMO platform. The main purpose of acceptance testing is to evaluate service compliance with user requirements and verify if the software is ready for deployment via NEMO's actual production environment. This procedure aims to identify any issues that may arise when deploying a real-world scenario. To address these issues, NEMO will perform several tests during the registration of each workload and collect the appropriate results. Next, the NEMO administrator could either accept the registered workload manually or define evaluation rules to accept the workload automatically. The acceptance methodology will involve the following steps:

a) Definition of the Acceptance Criteria: Definition of the acceptance criteria that match the user requirements and expectations and the compatibility with the NEMO platform.
b) Creation of the Test Cases: Development of automated pipelines that will execute well-documented use cases to ensure the comprehensive testing procedure.
c) Testing Execution: Execution of the test cases in an environment that closely resembles the production environment.
d) Test Results analysis: Collection and evaluation of the test results and identification of any defects or issues that need to be addressed.
e) Acceptance approval: Approval for production deployment.

4.2 Syntax testing

In NEMO, we introduced a syntax testing approach that aims to catch errors early in the deployment process of the workloads, reducing the likelihood of issues during the actual deployment. This process ensures that the descriptors, configuration files, and scripts used to deploy the workload are syntactically correct and can be successfully executed. Furthermore, during the syntax testing, NEMO will ensure that the registered workload is correctly configured and provides all the necessary information that is required by the NEMO platform not only for a successful deployment but also for the usage of the specific features that NEMO offers like High Availability, Accountability, and Monetization, Intent-based deployment and migration, etc. In detail the syntax testing procedure will include:

a) Configuration File Validation. Examination of all configuration files and scripts that are associated with the NEMO workload such as property files, YAML files, or JSON files, to ensure they are properly structured and adhere to the expected syntax.
b) Integrated Validation. Verification of the entire deployment process, including the execution of scripts and the application of configuration settings, can be completed without any syntax-

related issues. For this reason, a full-stack validation will be performed by running the deployment process in V&V environment.

4.2.1 Performance testing

NEMO performance evaluation as system focuses is features like speed, responsiveness, stability, and resource utilization, under various load conditions. This type of testing aims to identify potential bottlenecks, optimize system performance, and ensure that NEMO meets the required performance criteria. In this context, special attention will be paid to NEMO API testing to verify their functionality, reliability, and performance. These tests will include:

a) Stress testing: Evaluation of the system behaviour under normal and extreme load conditions.
b) Soak testing: Evaluation of the system's behaviour and resource consumption over an extended period.
c) Spike testing: Evaluation of the system's behaviour in case of sudden and large increases in load.

4.2.2 Security testing

NEMO provides specific security features supporting the deployment of secured workload over the continuum. For this reason, NEMO integrates well-known open-source tools and frameworks according to the SAST/DAST paradigm that evaluate each workload before the deployment in production. The appropriate tests are executed either as part of the registration process or in the V&V environment in which each workload can be deployed and tested in an isolated and safe sandbox. The security features that can be deployed by NEMO are:

a) Secure Code Review: Code analysis tools to scan the codebase for known security patterns and potential vulnerabilities. NEMO scans container images, file systems, and configuration files (such as Kubernetes) for known vulnerabilities.
b) Penetration Testing: Simulation of real-world attacks to evaluate the ability to withstand and detect malicious activities.
c) Compliance Checks: NEMO will integrate open-source frameworks to check compliance standards, such as CIS Benchmarks, HIPAA, and GDPR, and can check for violations within container images, file systems, and IaC configurations.
d) Dependency Tracking: Analysis of the dependencies within container images, file systems, and NEMO configurations files to identify known dependencies.

4.2.3 Functional testing

NEMO through V&V, will support functionality testing feature to check specific a slice of functionality in a workload focusing on particular behaviours or even a complex one composed by many units micro-services over the continuum. The purpose of this testing approach is to evaluate test whether the expected behaviour is successfully done by the system rather than the internal structure or implementation.

4.3 Open-source frameworks for testing

NEMO V&V will provide a well-structured framework, as part of the DevOps approach, that will facilitate several test types for each new workload from the development, integration, and deployment phases. This ensures that each new service will fulfil the requirements, Key Performance Indicators (KPIs) and is compatible with the innovative features that the NEMO platform offers (i.e., resource scaling, high availability, full-stack automated operations, etc.). One of the essential characteristics of the NEMO V&V framework is modularity, meaning that the system should be flexible enough to integrate new services and test tools easily. Considering the heterogeneity of modern network services, each service requires different testing approaches and tools. For this reason, NEMO will integrate several open-source testing tools and frameworks to address the different needs of each service and provide flexibility to developers to choose or introduce the framework of their choice as they know

better the needs of their implementation. Next, we present the open-source state-of-the-art frameworks that are already used by the industry and will be integrated into NEMO.

4.3.1 Selenium

Selenium [41] is a testing framework that allows web application testing across multiple web browser platforms and supports multiple modern programming languages. It is an umbrella project for a range of tools and libraries that enable and support the automation of web browsers by providing extensions to emulate user interaction with browsers, a distribution server for scaling browser allocation, and the infrastructure for implementations of the W3C WebDriver specification that lets you write interchangeable code for all major web browsers. Selenium brings together browser vendors, engineers, and enthusiasts into an ecosystem for the automation of web application testing and development. NEMO Integrates Selenium tests into the CI/CD pipelines to enhance the performance of the development cycles. So, every code change triggers a new build, followed by unit tests, integration testing, and Selenium-based automated tests.

```
default:
  image: node:8.10

stages:
  - deploy
  - confidence-check

test_nemo:
  stage: deploy
  script:
    nemo-deploy.sh test-workload
    - echo
  environment: NEMO-VnV

e2e:firefox:
  stage: confidence-check
  services:
    - selenium/standalone-firefox
  script:
    - npm run confidence-check --host=selenium__standalone-firefox

e2e:chrome:
  stage: confidence-check
  services:
    - selenium/standalone-chrome
  script:
    - npm run confidence-check --host=selenium__standalone-chrome
```

Figure 27: CI/CD pipeline for automated testing based on Selenium

4.3.2 Trivy

Trivy [42] is an open-source and comprehensive security scanner for container images, file systems, git repositories, VMs, K8s clusters, etc. It is designed to provide an easy and efficient way to identify and address vulnerabilities in software artifacts. It can identify dependencies between OS packages and software implementations, known vulnerabilities, IaC issues and miss configurations, sensitive information and secrets, software licenses, etc.

```
> docker run aquasec/trivy image python:3.4-alpine -q

python:3.4-alpine (alpine 3.9.2)
================================
Total: 37 (UNKNOWN: 0, LOW: 4, MEDIUM: 16, HIGH: 13, CRITICAL: 4)

+---------+-----------------+----------+--------+-------------------+---------------+--------------------------------------------------------------+
| Library | Vulnerability   | Severity | Status | Installed Version | Fixed Version |                            Title                             |
+---------+-----------------+----------+--------+-------------------+---------------+--------------------------------------------------------------+
| expat   | CVE-2018-20843  | HIGH     | fixed  | 2.2.6-r0          | 2.2.7-r0      | expat: large number of colons in input makes parser consume  |
|         |                 |          |        |                   |               | high amount ...                                              |
|         |                 |          |        |                   |               | https://avd.aquasec.com/nvd/cve-2018-20843                   |
|         +-----------------+          |        |                   +---------------+--------------------------------------------------------------+
|         | CVE-2019-15903  |          |        |                   | 2.2.7-r1      | expat: heap-based buffer over-read via crafted XML input     |
|         |                 |          |        |                   |               | https://avd.aquasec.com/nvd/cve-2019-15903                   |
+---------+-----------------+----------+        +-------------------+---------------+--------------------------------------------------------------+
| libbz2  | CVE-2019-12900  | CRITICAL |        | 1.0.6-r6          | 1.0.6-r7      | bzip2: out-of-bounds write in function BZ2_decompress        |
|         |                 |          |        |                   |               | https://avd.aquasec.com/nvd/cve-2019-12900                   |
+---------+-----------------+----------+        +-------------------+---------------+--------------------------------------------------------------+
| libcrypto1.1 | CVE-2019-1543 | HIGH |        | 1.1.1a-r1         | 1.1.1b-r1     | openssl: ChaCha20-Poly1305 with long nonces                  |
|         |                 |          |        |                   |               | https://avd.aquasec.com/nvd/cve-2019-1543                    |
|         +-----------------+          |        |                   +---------------+--------------------------------------------------------------+
|         | CVE-2020-1967   |          |        |                   | 1.1.1g-r0     | openssl: Segmentation fault in SSL_check_chain causes denial |
|         |                 |          |        |                   |               | of service                                                   |
|         |                 |          |        |                   |               | https://avd.aquasec.com/nvd/cve-2020-1967                    |
|         +-----------------+          |        |                   +---------------+--------------------------------------------------------------+
|         | CVE-2021-23840  |          |        |                   | 1.1.1j-r0     | openssl: integer overflow in CipherUpdate                    |
|         |                 |          |        |                   |               | https://avd.aquasec.com/nvd/cve-2021-23840                   |
|         +-----------------+          |        |                   +---------------+--------------------------------------------------------------+
|         | CVE-2021-3450   |          |        |                   | 1.1.1k-r0     | openssl: CA certificate check bypass with                    |
|         |                 |          |        |                   |               | X509_V_FLAG_X509_STRICT                                      |
|         |                 |          |        |                   |               | https://avd.aquasec.com/nvd/cve-2021-3450                    |
+---------+-----------------+----------+--------+-------------------+---------------+--------------------------------------------------------------+
```

Figure 28: Vulnerability report by Trivy

4.3.3 Syft

Syft [43] is a powerful and easy-to-use open-source tool for generating Software Bill of Materials (SBOMs) for container images and filesystems. It provides detailed visibility into the packages and dependencies in your software, helping you manage vulnerabilities, license compliance, and software supply chain security. It can be used as a standalone tool or it can be combined with Grype [44], and supports OCI, Docker and Singularity image formats. It can be easily integrated into CI/CD pipelines as it supports execution on several OS (i.e. Alpine, Debian, Redhat), programming languages (i.e. C/C++, Go, Java, Python, etc.), and frameworks (i.e. Jenkins).

```
> docker run anchore/syft python:3.4-alpine --scope all-layers
NAME                    VERSION          TYPE
.python-rundeps         0                apk
alpine-baselayout       3.1.0-r3         apk
alpine-keys             2.1-r1           apk
apk-tools               2.10.3-r1        apk
busybox                 1.29.3-r10       apk
ca-certificates         20190108-r0      apk
ca-certificates-cacert  20190108-r0      apk
expat                   2.2.6-r0         apk
gdbm                    1.13-r1          apk
libbz2                  1.0.6-r6         apk
libc-utils              0.7.1-r0         apk
libcrypto1.1            1.1.1a-r1        apk
libffi                  3.2.1-r6         apk
libressl2.7-libcrypto   2.7.5-r0         apk
libressl2.7-libssl      2.7.5-r0         apk
libssl1.1               1.1.1a-r1        apk
libtls-standalone       2.7.4-r6         apk
musl                    1.1.20-r3        apk
musl                    1.1.20-r4        apk
musl-utils              1.1.20-r3        apk
musl-utils              1.1.20-r4        apk
ncurses-libs            6.1_p20190105-r0 apk
ncurses-terminfo        6.1_p20190105-r0 apk
ncurses-terminfo-base   6.1_p20190105-r0 apk
pip                     19.0.3           python
python                  3.4.10           binary
readline                7.0.003-r1       apk
scanelf                 1.2.3-r0         apk
setuptools              40.8.0           python
sqlite-libs             3.26.0-r3        apk
ssl_client              1.29.3-r10       apk
wheel                   0.33.1           python
xz-libs                 5.2.4-r0         apk
zlib                    1.2.11-r1        apk
```

Figure 29: Automated Software Bill of Materials (SBOMs) by Syft

4.3.4 Grype

Grype [44] framework is a powerful and versatile vulnerability scanning tool that helps organizations improve the security and compliance of their software systems. It is designed for speed and efficiency, with the ability to scan container images and file systems, even for large and complex software artifacts.

This framework can easily be integrated into CI/CD pipelines, scan software artifacts, prioritize vulnerabilities, and track recommendation efforts. It can be also used to perform compliance checks with industry-specific standards, such as CIS Benchmarks, HIPAA, and GDPR, helping organizations ensure that their software artifacts adhere to relevant regulations and best practices. Overall, Grype provides visibility into the security of software components, including dependencies, to better manage software supply chain risks.

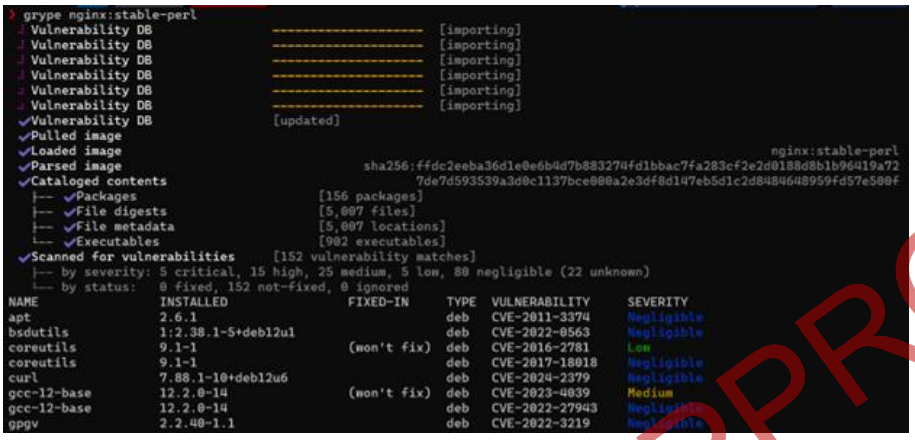


Figure 30: Vulnerability report of container image by Grype

5 Conclusions

The present document has provided the final architectural specifications of the NEMO metaOS. The document updates the initial NEMO metaOS meta-architecture, instantiated in D1.2.

Through the proposed metaOS meta-architecture, NEMO aims to facilitate the design and development of higher-level (meta) operating systems for the smart Internet of Things with strong computing capacity at the smart device, system and edge-level, embedded in a compute continuum from IoT-to-edge-to-cloud. The final version of the NEMO meta-architecture specifies the architectural views defined in the Meta-Architecture Framework (MAF) for the design of the NEMO metaOS.

The document clarifies the Network, User, Operational, Functional and Development views. In contrast, the Logical, Process and Physical views realize updates or further extensions. The updates incorporate development and integration options and needs, as well as feedback from Living Lab users. The specified architecture is aligned to the reference architecture of the EUCEI community at functional and component level. The outcome of this comparative analysis and mapping indicate that NEMO addresses the full-fledged functionality of metaOS, as perceived by the EUCloudEdgeIoT community, realizing a strong representation of European research and innovation teams across Europe.

Moreover, the document presents the NEMO Validation & Verification (V&V), with guided reference to testing approaches, as well as appropriate validation tools, to be considered during the project's verification and validation activities.

6 References

[1] NEMO, "D1.2 - NEMO meta-architecture, components and benchmarking. Initial version," HORIZON - 101070118 - NEMO Deliverable Report, 2023.

[2] P. Neuenschwander and S. Michelakis, "The infestation of Dacus oleae (Gmel.) (Diptera, Tephritidae) at harvest time and its influence on yield and quality of olive oil in Crete," *Zeitschrift für Angewandte Entomologie,* vol. 86, 1978.

[3] V. Vizzarri, L. Lombardo, C. Novellis, P. Rizzo, M. Pellegrino, G. Cruceli, G. Godino, F. Zaffina and A. Ienco, "Testing the Single and Combined Effect of Kaolin and Spinosad against Bactrocera oleae and Its Natural Antagonist Insects in an Organic Olive Grove," *Life,* 2023.

[4] H. Chen, Y. Lan, B. Fritz, W. C. Hoffmann and S. Liu, "Review of agricultural spraying technologies for plant protection using unmanned aerial vehicle (UAV)," *International Journal of Agricultural ans Biological Engineering,* vol. 14, no. 1, 2021.

[5] D. Tsouros, S. Bibi and P. Sarigiannidis, "A Review on UAV-Based Applications for Precision Agriculture," *Information,* vol. 10, no. 11, 2019.

[6] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision Agriculture,* 2012.

[7] A. A. Ayranci and B. Erkmen, "Edge Computing and Robotic Applications in Modern Agriculture," in *2024 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2024.

[8] Z. Z. Al-Mashhadani and J.-H. Park, "Autonomous Agricultural Monitoring Robot for Efficient Smart Farming," in *2023 23rd International Conference on Control, Automation and Systems (ICCAS)*, 2023.

[9] S. R. L, J. J, A. S, A. Mallick and B. V, "Soil Moisture Monitoring and Seed Sowing Robot with ThingSpeak Integration," in *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, 2024.

[10] W. Jiang, L. Quan, G. Wei, C. Chang and T. Geng, "A conceptual evaluation of a weed control method with post-damage application of herbicides: A composite intelligent intra-row weeding robot," *Soil and Tillage Research,* vol. 234, 2023.

[11] C.-H. Wang, Q.-K. Pan, X.-P. Li, H.-Y. Sang and B. Wang, "A multi-objective teaching-learning-based optimizer for a cooperative task allocation problem of weeding robots and spraying drones," *Swarm and Evolutionary Computation,* vol. 87, 2024.

[12] NEMO, "D1.1 - Definition and analysis of use cases and GDPR compliance," HORIZON - 101070118 - NEMO Deliverable Report, 2023.

[13] NEMO, "D5.2 - Living Labs and Data Management Plan (DMP). Final version," HORIZON - 101070118 - NEMO Deliverable Report, 2024.

[14] ISO, "ISO/IEC/IEEE 42010:2022 Software, systems and enterprise — Architecture description," 2022. [Online]. Available: https://www.iso.org/standard/74393.html. [Accessed 2023].

[15] M. Anisetti, F. Berto and M. Banzi, "Orchestration of data-intensive pipeline in 5G-enabled Edge Continuum," in *2022 IEEE World Congress on Services (SERVICES)*, Barcelona, Spain, 2022.

[16] F. Tusa and S. Clayman, "End-to-end slices to orchestrate resources and services in the cloud-to-edge continuum," *Future Generation Computer Systems,* vol. 141, 2023.

[17] I. Čilić, I. P. Žarko, and M. Kušek, "Towards Service Orchestration for the Cloud-to-Thing Continuum," in *6th International Conference on Smart and Sustainable Technologies (SpliTech)*, Split and Bol, Croatia, 2021.

[18] I. Tzanettis, C.-M. Androna, A. Zafeiropoulos, E. Fotopoulou and S. Papavassiliou.

[19] A. Orive, A. Agirre, H.-L. Truong, I. Sarachaga and M. Marcos, "Quality of Service Aware Orchestration for Cloud–Edge Continuum Applications," *Sensors,* vol. 22, no. 5, 2022.

[20] A. N. Al-Quzweeni, A. Q. Lawey, T. E. H. Elgorashi and J. M. H. Elmirghani, "Optimized Energy Aware 5G Network Function Virtualization," *IEEE Access,* vol. 7, 2019.

[21] A. Leivadeas and M. Falkner, "Autonomous Network Assurance in Intent Based Networking: Vision and Challenges," in *32nd International Conference on Computer Communications and Networks (ICCCN)*, 2023.

[22] A. Raghunandan, D. Kalasapura and M. Caesar, "Digital Twinning for Microservice Architectures," in *ICC 2023 - IEEE International Conference on Communications*, 2023.

[23] Z. Anastasakis, T.-H. Velivassaki, A. Voulkidis, S. Bourou, K. Psychogyios, D. Skias and T. Zahariadis, "FREDY: Federated Resilience Enhanced with Differential Privacy," *Future Internet,* vol. 15, no. 9, 2023.

[24] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow and K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," *arxiv.*

[25] International Data Spaces Association, "IDSA Reference Architecture Model," 2022.

[26] O. Segou, D. Skias, T. H. Velivassaki, T. Zahariadis, E. Pages, R. Ramiro, R. Rossini, P. Karkazis, A. M. D. Costa, L. M. C. Murillo, A. D. Rio, J. Serrano and Jimenez, "NExt generation Meta Operating systems (NEMO) and Data Space: envisioning the future," in *eSAAM'24 on Data Spaces*.

[27] FIWARE, "Orion Context Broker," [Online]. Available: https://github.com/telefonicaid/fiware-orion/. [Accessed 2024].

[28] FIWARE, "IoT Agents," FIWARE Academy, [Online]. Available: https://fiware-academy.readthedocs.io/en/latest/iot-agents/idas.html. [Accessed 2024].

[29] FIWARE, [Online]. Available: https://www.fiware.org. [Accessed 2024].

[30] FIWARE, "Build your own IoT platform with FIWARE enablers," 20215. [Online]. Available: https://www.fiware.org/2015/03/27/build-your-own-iot-platform-with-fiware-enablers/. [Accessed 2024].

[31] S. Sotiriadis, K. Stravoskoufos, E. G. Petrakis, V. Angelakis, E. Tragos, H. Pöhls, A. Kapovits and A. Bassi, "Future Internet Systems Design and Implementation: Cloud and IoT Services Based on IoT-A and FIWARE," in *esigning, Developing, and Facilitating Smart Cities: Urban Design to IoT Solutions*, Cham, Springer International Publishing, 2017, pp. 193-207.

[32] FIWARE, "Knowage and NGSI," FIWARE Knowage, [Online]. Available: https://knowage.readthedocs.io/en/6.1.1/user/NGSI/README/index.html. [Accessed 2024].

[33] IDSA, "IDS Connector," [Online]. Available: https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/layers-of-the-reference-architecture-model/3-layers-of-the-reference-architecture-model/3_5_0_system_layer/3_5_2_ids_connector. [Accessed 2024].

[34] EC, "Future European platforms for the Edge: Meta Operating Systems (RIA)," [Online]. Available: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/horizon-cl4-2021-data-01-05. [Accessed 2024].

[35] EUCloudEdgeIoT.eu, "Building the European Cloud, Edge & IoT Continuum for business and research," European Commission Horizon Europe - 101070030 - 101070571, [Online]. Available: https://eucloudedgeiot.eu/. [Accessed 2024].

[36] EUCEI, "Developing a Reference Architecture for the Continuum - Concept, Taxonomy and Building Blocks," EC HORIZON Europe - 101070030 - OpenContinuum Report, 2023.

[37] EUCEI, "Compositional View of the Continuum Reference Architecture: Graphical representation of common and potential capabilities," EC Horizon Europe, 2024.

[38] NEMO, "D3.1 - Introducing NEMO Kernel," HORIZON - 101070118 - NEMO Deliverable Report, 2023.

[39] NEMO, "D2.1 Analysis Nemo Underlying Technology," HORIZON - 101070118 - NEMO Deliverable Report, 2023.

[40] UNLOCK-CEI, "Technology Implementation Model and Architectural Patterns for CEI Use Cases," HORIZON - 101070571 - UNLOCK-CEI, 2024.

[41] Selenium, "Selenium," [Online]. Available: https://www.selenium.dev/. [Accessed 2024].

[42] Aqua Security Software, "Trivy," [Online]. Available: https://trivy.dev/. [Accessed 2024].

[43] Anchore, "Syft," [Online]. Available: https://github.com/anchore/syft. [Accessed 2024].

[44] Anchore, "Grype," [Online]. Available: https://github.com/anchore/grype/. [Accessed 2024].

[45] B. Keith , "Near, Far or Tiny: Defining and Managing Edge Computing in a Cloud Native World," 2021. [Online]. Available: https://vmblog.com/archive/2021/04/27/near-far-or-tiny-defining-and-managing-edge-computing-in-a-cloud-native-world.aspx.

[46] Canonical, "MicroK8s - The lightweight Kubernetes," 2023. [Online]. Available: https://microk8s.io/.

[47] Rancher Labs, "K3s - Lightweight Kubernetes," 2023. [Online]. Available: https://k3s.io/.

[48] Mirantis, "K0s," 2023. [Online]. Available: https://k0sproject.io/.

[49] The Kubernetes Authors, "minikube," 2023. [Online]. Available: https://minikube.sigs.k8s.io/docs/.

[50] The KubeEdge Project Authors, "KubeEdge," 2023. [Online]. Available: https://kubeedge.io/.

[51] "Checkov homepage," BridgeCrew, [Online]. Available: https://www.checkov.io. [Accessed 25 Feb. 2022].