TRUSTWORTHY, COGNITIVE AND AI-DRIVEN COLLABORATIVE ASSOCIATIONS OF IOT DEVICES AND EDGE RESOURCES FOR DATA PROCESSING

Grant Agreement no. 101136024

Deliverable D2.1
State of the art, use cases analysis, platform requirements and KPIs

Programme:	HORIZON-CL4-2023-DATA-01-04
Project number:	101136024
Project acronym:	EMPYREAN
Start/End date:	01/02/2024 – 31/01/2027

Deliverable type:	Report
Related WP:	WP2
Responsible Editor:	UMU
Due date:	31/07/2024
Actual submission date:	31/07/2024 (initial submission v. 1.3) 24/10/2025 (revised version)

Dissemination level:	Public
Revision:	FINAL

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101136024

Revision History

Date	Editor	Status	Version	Changes
12/07/2024	Eduardo Cánovas, Antonio Skarmeta (UMU)	Draft	1.0	First complete version for internal review.
17/07/2024	Roberto González	Draft	1.1	First internal review
19/07/2024	Aristotelis Kretsis, Panagiotis Kokkinos	Draft	1.2	Second internal review
26/07/2024	Partners inputs	Draft	1.2	Last inputs from partners
29/07/2024	Eduardo Cánovas, Antonio Skarmeta (UMU)	Final	1.3	Final check
24/10/2025	Eduardo Cánovas, Alonso Sánchez	Revised	1.4	Address first project review comments

Author List

Organization	Author
ICCS	Aristotelis Kretsis, Panagiotis Kokkinos, Emmanouel Varvarigos
NVIDIA	Dimitris Syrivelis
CC	Márton Sipos, Marcell Fehér
UMU	Antonio Skarmeta, Eduardo Cánovas
ZSCALE	Iván Paez
RYAX	Yuqiang Ma, Yiannis Georgiou
NUBIS	Christos Panagiotou, Anastasios Tsakas, Anastassios Nanos, Ilias Lagomatis, Kostis Papazafeiropoulos
IDEKO	Javier Martín, Aitor Fernández
NEC	Roberto González, Jaime Fúster
EV ILVO	Theodoros Chalazas, Panagiotis Ilias
TRAC	Keshav Chintamani

Internal Reviewers

Aristotelis Kretsis, Polyzois Soumplis, ICCS

Roberto González, NEC

Abstract: This deliverable summarized the project analysis for the state of the art related to the technologies being considered, an extensive use cases analysis, and finally, based on this the identification of the platform requirements and associated KPIs. The deliverable presents the outcomes of Task 2.1 “State-of-the-Art Analysis” and Task 2.2 “Concept, Use Cases and Requirements Analysis” as part of the Work Package 2 “Use Cases Analysis, System Requirements and Overall Architecture” of the EMPYREAN project, during its first iteration. It includes EMPYREAN evaluation of state-of-the-art and beyond that will be covered in the project for the different technologies being advanced within the project’s activities, the detailed specification of the EMPYREAN use cases, and the initial analysis of the functional and non-functional requirements for the EMPYREAN components. The deliverable additionally reviews the enablers being considered for providing secure identity management, real-time monitoring, efficient data transport, and intelligent workload management. They address platform challenges, enabling seamless orchestration, robust security, and optimal resource utilization across various applications and use cases.

Keywords: State-of-the-art, use case definition, functional requirements analysis, EMPYREAN concept, EMPYREAN enablers

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ATR	Autonomous Towing Robots
AWS	Amazon Web Services
CNC	Computer Numerical Control
CNCF	Cloud Native Computing Foundation
CNN	Convolutional Neural Network
CPS	Cyber Physical Systems
CTA	Cyber Threat Alliance
CTI	Cyber Threat Intelligence
CVEs	Common Vulnerabilities and Exposures
D	Deliverable
DDS	Data Distribution Service
DID	Decentralized Identifiers
DKMA	Distributed Key Management and Authentication
DLT	Distributed Ledger Technology
DNN	Deep Neural Network
DoA	Description of Action
DQN	Deep Q-Learning
DRL	Deep Reinforcement Learning
DTR	Decision Tree Regression
EC	European Commission
EO	Earth Observation
EUs	End Users
FCS	Fleet Control System
FL	Federated Learning
FOA	Fog Orchestrator Agent
FPGA	Field Programmable Gate Arrays
GIS	Geographic Information System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HPC	High Performance Computing
HW	Hardware
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
ILP	Integer Linear Programming
IoC	Indicators of Compromise
IoT	Internet of Things
IPC	Industrial PC
K8s	Kubernetes
KMS	Key Management System
KPI	Key Performance Indicator
LAN	Local Area Network
LSTM	Long Short-Term Memory

MANO	Management and Orchestration
MEC	Mobile Edge Cloud
ML	Machine Learning
MLP	Mixed Linear Programming
MQTT	Message Queueing Telemetry Transport
MTTR	Mean Time to Repair
NBS	Nash Bargaining Solution
NFV	Network Functions Virtualization
NIC	Network Interface Card
NIR	Near-Infrared Spectrum
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
OCI	Open Container Initiative
OTX	Open Threat Exchange
P2P	Peer-to-Peer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PLC	Programmable Logic Controller
PM	Project Manager
PO	Project Officer
PPFL	Privacy-Preserving Federated Learning
PSMS	Pointcheval–Sanders Multi-Signatures
PSO	Particle Swarm Optimization
QoS	Quality of Service
R2X	Robot to Everything
RDMA	Remote Direct Memory Access
REST	REpresentational State Transfer
RL	Reinforcement Learning
RLNC	Random Linear Network Coding
ROI	Return on Investment
SAM	Segment Anything
SIG	Special Interest Group
SLA	Service Level Agreement
SOC	Soil Organic Carbon
SSI	Self-Sovereign Identity
SW	Software
TPU	Tensor Processing Unit
UAV	Unmanned Aerial Vehicles
UC	Use Case
VC	Verifiable Credentials
Vis-NIR	Visible and Near-Infrared Spectrum
VM	Virtual Machine
VRA	Variable Rate Application
WAN	Wide Area Network
XACML	Extensible Access Control Markup Language
ZKP	Zero-Knowledge Proofs

1 Executive Summary

The EMPYREAN project aims to revolutionize the IoT-edge-cloud continuum by introducing a collaborative, cognitive, and trust-enhanced infrastructure. This deliverable provides a comprehensive state-of-the-art analysis across various domains such as security, trust, seamless data management, and decentralized intelligence. EMPYREAN advances beyond current technologies by developing a distributed trust management framework, enhancing secure shared data management through blockchain integration, providing novel application development and deployment mechanisms, enabling distributed decision-making, and optimizing AI-based workload autoscaling. The project's innovative approach includes collaborative continuum management, AI-enabled application deployment, and advanced data interconnection management. These advancements promise significant improvements in efficiency, security, and reliability for hyper-distributed applications across diverse and impactful application areas, such as smart agriculture, advanced manufacturing, and robot-based logistics.

Based on this innovative approach, EMPYREAN envisions a new paradigm for the continuum, introducing the concept of collaborative collectives of IoT devices, robots, and resources spanning from the edge to the cloud. EMPYREAN calls this the Association-based continuum, in the sense that multiple Associations (collaborative collectives of IoT devices, robots, and resources) operate in parallel in space and time and constitute the IoT-edge-cloud continuum. Each Association is composed of shared and aggregated edge computing and storage resources of various sizes and characteristics, encompassing both general-purpose and specialized units. These Associations are dynamically formed and updated based on the resource owners' participation, while central cloud resources can be utilized when and if needed. This approach promises to enhance the flexibility, efficiency, and collaboration within the IoT-edge-cloud ecosystem.

Within the deliverable, the detailed specification of the EMPYREAN use cases and the initial analysis of the functional and non-functional requirements for the EMPYREAN components and their associated baseline of enablers are being analysed.

The outcome of this deliverable will be used as a guideline for the architecture specification (D2.2), research activities regarding the EMPYREAN platform (WP3-5) and EMPYREAN Use Cases integration, development and evaluation (WP6), to focus on overcoming the current identified technical and scientific boundaries.

2 Introduction

2.1 Purpose of this document

The objective of this deliverable is to present the state-of-the-art analysis regarding the different technologies being advanced with the project's activities, as well as existing technologies, standards, and outcomes from ongoing relevant EU projects. The deliverable will review the approach of EMPYREAN in relation to technologies like identity, security, interoperability, resource and service orchestration, machine learning, extreme scale analytics, and cloud securitization.

The deliverable also defines and analyses the Use Cases (UCs) associated with the project evaluation and demonstration scenarios along with the collection of associated requirements. These requirements will serve as the basis for designing the platform's services, involving a comprehensive specification of the EMPYREAN platform as well as the requirements of each different building block; that latter will be detailed in the architecture defined in D2.2 (M7). Task 2.2 will also provide the means to identify, define, and quantify the Key Performance Indicators (KPIs) used to evaluate the project's innovations. Depending on the individual UC requirements, specific metrics for performance, interoperability, usability, and security, among others, will be targeted.

The final deliverable (D2.3) of WP2, scheduled for M12, will present a detailed analysis of the project use cases, the final analysis of both functional and non-functional requirements, as well as the main identified success criteria and the definition of the KPIs for evaluating the developed innovations.

2.2 Audience

This document is publicly available and should be of use to anyone interested in the detailed specification of the EMPYREAN project and its UCs, along with the EMPYREAN platform requirements analysis.

3 EMPYREAN State of the art

3.1 EMPYREAN Advances over state of the art

EMPYREAN aims to introduce and develop an ecosystem of innovative technologies and methodologies towards a collaborative, trustful and cognitive IoT-edge-cloud continuum. In this section, we present an overall view of the innovations that EMPYREAN pursues in different key areas.

3.1.1 Security, Trust, and Seamless Data and Computing Management

3.1.1.1 Collaborative Continuum's Management

The shift from a centralized computing model to a distributed one due to increasing mobile edge and IoT traffic is reflected in the infrastructure management and orchestration (MANO) model. Current MANO systems typically employ either (i) a centralized orchestrator hosted at a central cloud node, that oversees and makes decision on the entirety of the network, (ii) a federated system of regional orchestrators whose area of jurisdiction comprises a subset of localized, connected nodes or (iii) a hybrid system, usually consisting of two layers, with a high-level orchestrator that is responsible for assigning application requests to local orchestrators that control a subset of infrastructure nodes, with each local orchestrator subsequently performing the necessary resource allocation to serve the applications' workload and data, optimizing a set of objectives[1].

There are a number of platforms in the area of IoT-Edge-Cloud, some of them focus more on the edge-cloud or IoT part, while others consider IoT, Edge and Cloud. These platforms enable real-time data processing and storage of IoT data in the edge while also reducing bandwidth costs and avoiding transfers, while supporting further processing in the cloud when needed. Orchestration/scheduling of the resources is always one of the key aspects of the related operations and platforms [2]. The cloud federation reference architecture by NIST [3] can be also considered as a base when designing the architecture of such a global continuum.

[1] A. Ullah et al., "Orchestration in the Cloud-to-Things compute continuum: taxonomy, survey and future directions," Journal of Cloud Computing, vol. 12, no. 1, p. 135, Sep. 2023, doi: 10.1186/s13677-023-00516-5.
[2] M. Schwarzkopf and P. Bailis, "Research for practice: cluster scheduling for datacenters," Commun. ACM, vol. 61, no. 5, pp. 50–53, 2018.
[3] R. B. Bohn, C. A. Lee, and M. Michel, "The NIST Cloud Federation Reference Architecture," Feb. 2020, Accessed: Apr. 16, 2021. [Online]. Available: https://www.nist.gov/publications/nist-cloud-federation-reference-architecture.

3.1.1.1.1 Cloud Providers

Major public cloud providers, such as Google, Microsoft and Azure offer such services/platforms. Azure IoT Edge and Azure IoT Hub [4] enable to scale out and manage an IoT solution from the cloud, deploying, running and monitoring containerized Linux workloads. Also, other Azure services like Azure Defender and Azure Stream Analytics provide security and analytics services. Azure IoT Edge is made up of three components (Figure 2):

- IoT Edge modules are containers that run Azure services, third-party services, or other code. The IoT Edge runtime runs on each IoT Edge device and manages the modules deployed to each device.
- The runtime sits on the IoT Edge device, and performs management and communication operations between downstream devices and an IoT Edge device, between modules on an IoT Edge device, and between an IoT Edge device and the cloud.
- A cloud-based interface enables also to remotely monitor and manage IoT Edge devices.

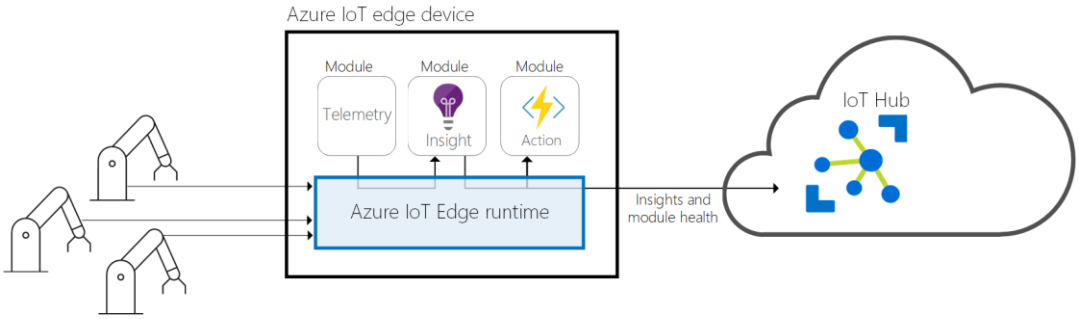

Figure 1: Azure IoT Edge and Azure IoT Hub.

Google Cloud IoT & IoT Core[5] is a fully managed service that allows to easily and securely connect, manage, and ingest data from millions of globally dispersed devices. Google Cloud IoT Core, in combination with other services on Google Cloud IoT platform, provides a complete solution for collecting, processing, analyzing, and visualizing IoT data in real time. In particular, it is possible to build and train ML models in the cloud, then run those models on the Cloud IoT Edge devices, while utilizing Edge TPU hardware accelerators and or GPU- and CPU-based accelerators. Edge TPU is Google's purpose-built ASIC chip designed to run TensorFlow Lite ML models at the edge. Cloud IoT Edge can run on Android Things or Linux OS-based devices, and its key components are (Figure 3):

- A runtime for gateway class devices, that provides local processing, while seamlessly interoperating with the rest of Cloud IoT platform.

[4] "Azure IoT Edge: Extend Cloud Intelligence and Analytics to Edge Devices," Microsoft Azure. Available: https://azure.microsoft.com/en-us/products/iot-edge

[5] "Google Cloud IoT Core: Managed Service for Connecting and Managing IoT Devices," Google Cloud. Available: https://cloud.google.com/iot-core

- The Edge IoT Core runtime that more securely connects edge devices to the cloud, enabling software and firmware updates and managing the exchange of data with Cloud IoT Core.
- The TensorFlow Lite-based Edge ML runtime that performs local ML inference using pre-trained models.

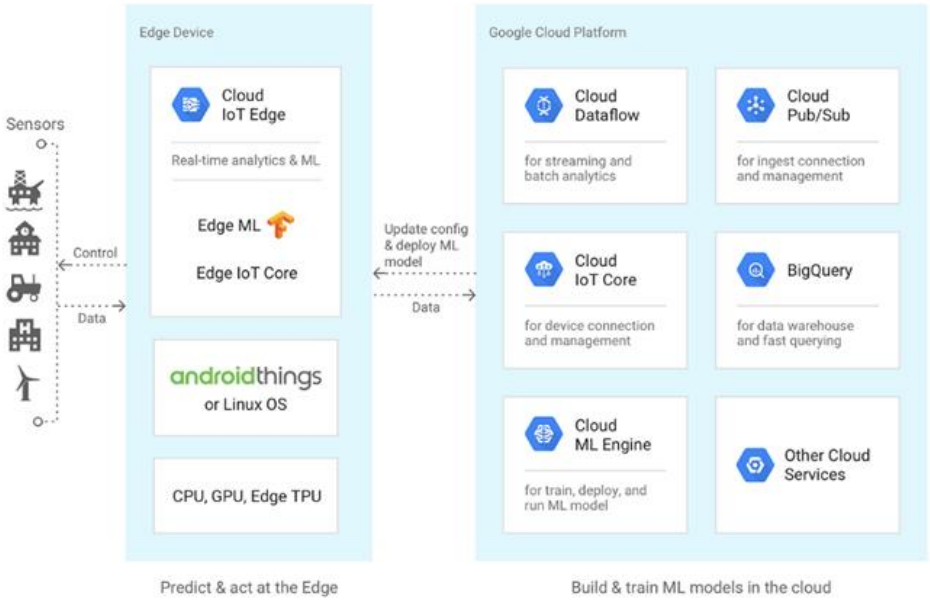

Figure 2: Google Cloud IoT & IoT Core and the execution of ML workloads.

AWS offers a variety of services for IoT and edge. AWS IoT Core is a managed cloud service that enables connected devices to securely interact with cloud applications and other devices. AWS IoT Greengrass is an IoT open source edge runtime that helps build, deploy, and manage device software. Amazon SageMaker enables developers to optimize machine learning (ML) models for inference in the cloud and supported devices at the edge. Figure 3 illustrates an example of deploying and using GPU accelerated image classification utilizing NVIDIA Jetson modules and NVIDIA's and AWS services.

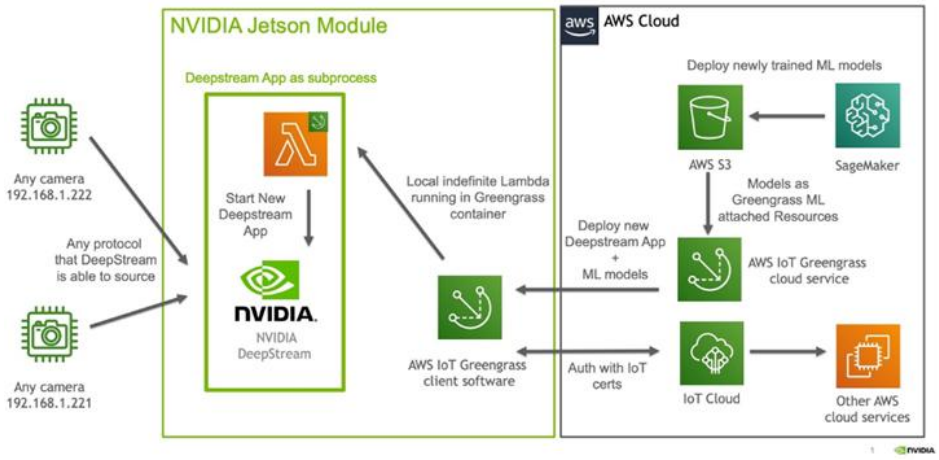

Figure 3: An example use of Amazon's IoT, Edge and ML related services.

3.1.1.1.2 Open Source and Other platforms

There are also a number of open source platforms that target the IoT-Edge-Cloud continuum.

As far as container and microservice orchestration in the cloud is concerned, Kubernetes (K8s) [6,7,8] has been the standard solution, simplifying deployment, and incorporating scheduling, scaling, and monitoring. Mesos [9] is also an advanced and known open-source orchestrator. Kubernetes orchestrator enables the support of Software Defined Infrastructures and resources disaggregation by leveraging on container-based deployments and particular drivers based on standardized interfaces (Container Runtime Interface [10], Container Storage Interface [11], Container Network Interface [12] and the device plugins framework [13]). These interfaces enable the definition of abstractions for finer-grain control of computation, state, and communications in multi-tenant environments along with optimal usage of the underlying hardware resources. However, even if Kubernetes is today production-level for typical cloud data centers, the default distribution is not adapted for the constrained and heterogeneous edge capabilities nor for multi-cluster deployments, so as to integrate different layers of compute resources (edge, fog and cloud).

Efforts are currently ongoing to better adapt Kubernetes for the edge, such as those done by the IoT-Edge working group [14] which are mainly focused on guidelines and best practices with the current ecosystem. Also, there have been several attempts at creating frameworks that can be utilized as K8s alternatives that are usable in edge-computing paradigms. These solutions can be divided into two main categories. The first school of thought attempts to replicate K8s by adopting the same centralized structure but also reducing the necessary system requirements for employing Kubernetes so that its deployment can be viable in a broader range of potential devices, while the second tries to create a novel framework that is better suited for distributed computing environments by utilizing a number of main Kubernetes principles but also adapting them to the specificities of edge and aiming for cross layer cooperation and integration [15].

[6] Kubernetes, "Overview." May 2023. [Online]. Available: https://kubernetes.io/docs/concepts/overview/
[7] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade," Queue, vol. 14, no. 1, pp. 70–93, 2016.
[8] "Production-Grade Container Orchestration," Kubernetes. https://kubernetes.io/ (accessed Apr. 16, 2021).
[9] B. Hindman et al., "Mesos: A platform for fine-grained resource sharing in the data center.," in NSDI, 2011, vol. 11, no. 2011, pp. 22–22.
[10] "Introducing Container Runtime Interface (CRI) in Kubernetes," Kubernetes, Dec. 19, 2016. https://kubernetes.io/blog/2016/12/Container-Runtime-Interface-Cri-In-Kubernetes/ (accessed Apr. 16, 2021).
[11] "Container Storage Interface (CSI) for Kubernetes GA," Kubernetes, Jan. 15, 2019. https://kubernetes.io/blog/2019/01/15/container-storage-interface-ga/ (accessed Apr. 16, 2021)
[12] containernetworking/cni. CNI, 2021
[13] "Device Plugins," Kubernetes. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/ (accessed Apr. 16, 2021).
[14] "kubernetes/community," GitHub. https://github.com/kubernetes/community (accessed Apr. 16, 2021).
[15] R. Vaño, I. Lacalle, P. Sowiński, R. S-Julián, and C. E. Palau, "Cloud-Native Workload Orchestration at the Edge: A Deployment Review and Future Directions," Sensors, vol. 23, no. 4, p. 2215, Feb. 2023, doi: 10.3390/s23042215.

In the first category, employing the centralized aggregation architecture of K8s, K3s [16] has seen a great rise in popularity. In k3s, Kubernetes heavyweight internal procedures have been stripped down as another alternative that simplifies the autonomy of single edge devices using the same Kubernetes API. Achieving a great reduction in minimum requirements, K3s can run on several different architectures which is ideal for employing clusters at the edge. Also in that category, published by Canonical as a lightweight alternative to K8s, MicroK8s [17] offers a clear performance improvement for low computing power and capacity devices to its parent. Microk8s can enable the deployment of individual autonomous edge resources, needing to orchestrate tasks and workflows autonomously when disconnected from the network. Both distributions enable cluster deployment at the edge but employ a central aggregator that cannot be easily expanded to multi-cluster scenarios due to the complexities in the cross-layer communication.

Regarding the second group, the deployment is focused on transferring the principles of Kubernetes to the edge, by keeping the control plane of the system in the cloud and implementing specific controllers and modules for the edge, allowing for varying degrees of autonomy on the application level. A typical example of a framework of that type is KubeEdge [18], which is an open-source project by the Cloud Native Computing Foundation (CNCF) and driven by several vendors. It facilitates seamless integration of edge nodes into Kubernetes clusters, allowing orchestration and application monitoring at the edge to a degree, while the cloud is responsible for high level decision-making on workload provisioning and reconfiguration. KubeEdge is built upon Kubernetes and extends native containerized application orchestration and device management to hosts at the Edge while being fully compatible with the Kubernetes APIs. It ensures that edge nodes run autonomously and the applications in edge run normally, when the cloud-edge network is unstable or edge is offline and restarted. It consists of cloud part and edge part and supports MQTT. With KubeEdge it is easy to get and deploy existing complicated machine learning, image recognition, event processing and other high level applications to the edge. KubeEdge also incorporates a cache mechanism on the edge to assure that no data is lost in case of connection loss.

The most important components of KubeEdge are the following (Figure 5):

- *CloudHub & EdgeHub*: a web socket server and client responsible for the edge cloud communication.
- *EdgeController*: an extended Kubernetes controller which manages edge nodes and pods metadata so that the data can be targeted to a specific edge node.
- *DeviceController*: an extended Kubernetes controller which manages devices so that the device metadata/status data can be synced between edge and cloud.
- *Edged*: an agent that runs on edge nodes and manages containerized applications.

[16] "K3s." [Online]. Available: https://k3s.io

[17] "MicroK8s - Zero-ops Kubernetes for developers, edge and IoT | MicroK8s." [Online]. Available: https://microk8s.io

[18] "KubeEdge, a Kubernetes Native Edge Computing Framework." May 2019. [Online]. Available: https://kubernetes.io/blog/2019/03/19/kubeedge-k8s-based-edge-intro/

- *EventBus*: a MQTT client to interact with MQTT servers (mosquitto), offering publish and subscribe capabilities to other components.

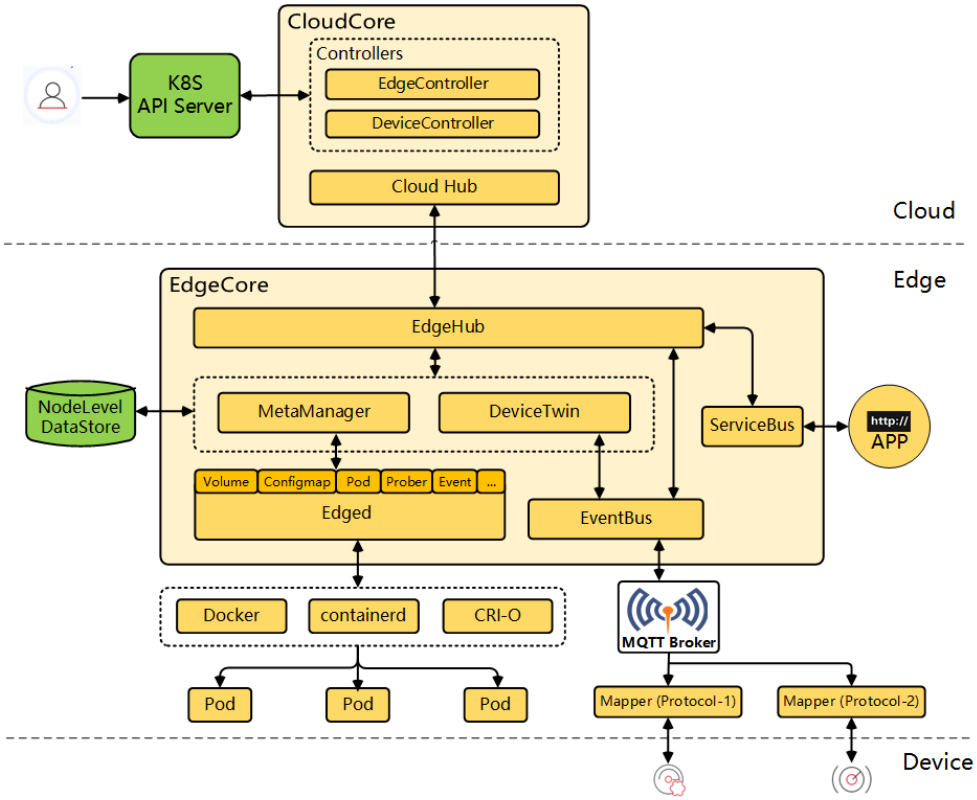

Figure 4: KubeEdge architecture.

In a similar way, the multi-cluster special interest group (SIG) community of Kubernetes works on the federation v2 project, named kubefed [19] that focuses on integrating multiple clusters under a federation while providing a generic scheduling engine that, based on policies, can make decisions on how to place arbitrary Kubernetes API objects. The project is now deprecated but other open source software such as Karmada[20] and Kubesphere[21] have adopted some advanced aspects of kubefed and can be currently used in production.

EdgeX Foundry [22] is a standardized interoperability framework for IoT edge computing. It can connect with various sensors and devices via different protocols, manage them and collect data from them, and export the data to a local application at the edge or the cloud for further processing. EdgeX is designed to be agnostic to hardware, CPU, operating system, and application environment. It can run natively or run in docker containers. EdgeX Foundry is a collection of open source micro services that are organized into 4 service layers, and 2 underlying augmenting system services (Figure 6):

[19] kubernetes-sigs/kubefed. Kubernetes SIGs, 2021
[20] Karmada, https://karmada.io/
[21] Kubeshpere, https://kubesphere.io/
[22] "EdgeX Foundry | The Open Source Edge Platform", [Online]. Available: www.edgexfoundry.org

- Application services are the means to extract, process/transform and send sensed data from EdgeX to an endpoint or process of your choice (e.g. major cloud providers, to MQTT(s) topics, and HTTP(s) REST endpoints)
- Supporting services layer include a wide range of microservices to perform edge analytics and other duties such as logging, scheduling, and data clean up.
- Core services layer acts as an intermediary between the device layer and application layer providing information about device connectivity, data flow and configurations.
- The device services are the edge connectors to interact with field level devices. The device service communicates with the devices, sensors, actuators, and other IoT objects through protocols native to each device object.
- Security elements of EdgeX Foundry protect the data and control of devices, sensors, and other IoT objects managed by EdgeX Foundry.
- System management service acts as a point of contact to external management systems to perform actions in EdgeX platform like START/ STOP/ RESTART and get metrics on used EdgeX services.

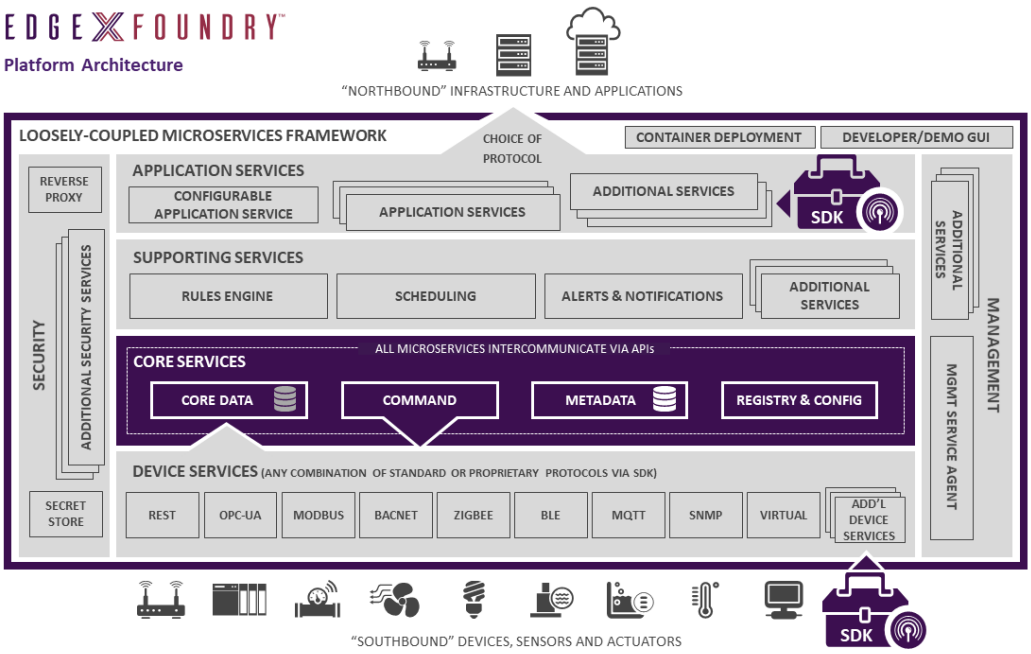

Figure 5: EdgeX Foundry architecture.

Similarly, OpenYurt [23] by Chinese giant Alibaba follows a typical cloud-edge architecture integrating more powerful edge data centers into Kubernetes clusters. Both implementations utilize secure tunnels to ensure privacy in heterogeneous device communication. Their traditional master-slave/worker design, however, offers limited usability in the case of edge isolation.

[23] "An open platform that extends upstream Kubernetes to Edge | OpenYurt." [Online]. Available: https://openyurt.io

Other federated frameworks like KubeFed [24], StarlingX [25] enable autonomy for subnetworks in case of outages. More specifically, Kubefed comprises a centralized server that distributes Kubernetes API objects to clusters, allowing for independent management. Similarly, StarlingX creates independent edge clouds, adopting a layered architecture that tasks a central cloud with an abstraction of the control plane and employing a number of sub-clouds. The main drawbacks of StarlingX and KubeFed lie in their inability to federate inter-cluster cooperation.

Other alternatives that could be used to provide the communication layer across services that run on multiple Kubernetes clusters are Submariner [26] and Istio[27] but they do not offer the capabilities of scheduling which has to be provided separately. Submariner utilizes a Broker object to orchestrate and synchronize the state information for all clusters to establish cooperation between Pods and Services belonging to different clusters. Submarines can handle scenarios where the connection to the cloud is lost, but any shifts in application requirements or anomalous events, cannot be solved locally as these require a -no longer available- higher-level decisions entity.

They are also commercial IoT-Edge-Cloud platforms. ClearBlade[28] is an edge computing software enabling enterprises to rapidly engineer and run secure, real-time, scalable industrial IoT applications.

3.1.1.1.3 More on Kubernetes Scheduler

The decisions regarding where to execute pods are taken by the Kubernetes schedulers. Kube-scheduler is the default scheduler of Kubernetes and its role is to dispatch pods to nodes based on a scheduling policy[29]. Depending on the metric focused by the platform, sometimes the best allocation is not achieved with such a scheduler. Besides Kubernetes allowing the developers to modify its default one, there are some available options such other open-source schedulers for Kubernetes such as Volcano[30], YuniKorn [31],[32], Safe Scheduler[33] and Multiple ClusterDispatcher[34]. Each one addresses some of the different focus on scheduling objectives. Since all the tools mentioned above are based on Kubernetes and its default scheduler, we will depict its mechanism.

The Kubernetes scheduling process is based on some steps where filtering and scoring are the two main parts. Basically, before scheduling a pod to a node, the kube-scheduler filters all

[24] "Kubernetes Cluster Federation (KubeFed)." May 2023. [Online]. Available: https://github.com/kubernetes-retired/kubefed
[25] "Open-Source Edge Cloud Computing Architecture - StarlingX." [Online]. Available: https://www.starlingx.io
[26] "Submariner (2022)." May 2024. [Online]. Available: https://github.com/submariner-io/submariner
[27] Istio, https://istio.io/
[28] "ClearBlade: Edge Computing Platform for Enterprise IoT", [Online]. Available: www.clearblade.com
[29] "Kubernetes Scheduler," Kubernetes. https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/ (accessed Apr. 01, 2021).
[30] "Volcano." https://volcano.sh/en/ (accessed Apr. 16, 2021).
[31] "Welcome to Apache YuniKorn (Incubating) | Apache YuniKorn (Incubating)." https://yunikorn.apache.org (accessed Apr. 16, 2021).
[32] apache/incubator-yunikorn-core. The Apache Software Foundation, 2021.
[33] IBM/kube-safe-scheduler. International Business Machines, 2021.
[34] IBM/multi-cluster-app-dispatcher. International Business Machines, 2021.

available nodes and if there are any, it scores them to select the most valuable one. If the kube-scheduler algorithm is not the most suitable option for some specific case, Kubernetes allows developers to change it. Policies[35] are the way that Kubernetes implements the filtering and scoring phases for the scheduler. Anyone can access and change its parameters. Furthermore, Kubernetes split its scheduling process in stages, for instance, QueueSort, Filter, Score, Bind, Reserve and others. Each Stage represents a scheduling step, and they are exposed by Extension Points. Extension Points behaviors are implemented by Plugins[36]. A Plugin can be used by one or more Extension Points. In the end, the set of Extension Points and its Plugins compose a Profile[37]. A Profile is a mechanism that allows developers to configure Plugins to implement different scheduling behaviors for different Stages. If not provided at pod creation, Kubernetes considers its default Profile, the default-scheduler. There is also the possibility to develop several Profiles and to use them within different pods as a multiple-scheduler mechanism [38]. For instance, an example is the QueueSort Extension Point. As an Extension Point it defines a scheduling Stage. It provides ordering functions to sort the pods in the schedule queue. To do so the QueueSort Extension Point uses the PrioritySort Plugin, which implements the default priority-based sorting. With many others, QueueSort Extension Point, and consequently the PrioritySortPlugin, compose the default-scheduler Profile. Since the Kubernetes environment evolves and changes over time, some resources might become under or over usage. For instance, a very simple example is if a pod fails, and it is duplicated for fault tolerant reasons. If the failed pod becomes available again the cluster would be running two instances of the same pod. For this and other reasons, there is a mechanism to avoid such under or over resources usage named descheduler[39]. The goal of the descheduler mechanism is to find pods that can be moved and evict them. This does not mean that the descheduler will replace the evicted pods, but if needed, that can be done by the scheduler itself.

3.1.1.1.4 Research oriented platforms and surveys

In the research literature there are also a number of surveys about the IoT-Edge-Cloud continuum covering various aspects like, taxonomy, algorithms and software [40].

Castellano et al. [41] adopted a distributed approach in which orchestration is implemented at the application level, with an object being instantiated every time a new application is initiated. The decisions made by the service defined orchestrator (SDA) are based on policies

[35] "Scheduling Policies," Kubernetes. https://kubernetes.io/docs/reference/scheduling/policies (accessed Apr. 16, 2021).
[36] kubernetes-sigs/scheduler-plugins. Kubernetes SIGs, 2021.
[37] "Scheduler Configuration," Kubernetes. https://kubernetes.io/docs/reference/scheduling/config (accessed Apr. 16, 2021).
[38]"Configure Multiple Schedulers," Kubernetes. https://kubernetes.io/docs/tasks/extend-kubernetes/configure-multiple-schedulers (accessed Apr. 16, 2021).
[39] kubernetes-sigs/descheduler. Kubernetes SIGs, 2021.
[40] Liu, Fang, et al. "A survey on edge computing systems and tools", Proceedings of the IEEE, 107.8, pp. 1537-1562, 2019.
[41]G. Castellano, F. Esposito, and F. Risso, "A Service-Defined Approach for Orchestration of Heterogeneous Applications in Cloud/Edge Platforms," IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1404–1418, Dec. 2019, doi: 10.1109/TNSM.2019.2941639.

provided by a behavior model which produces its output based on system parameters such as the topology, the application deployment request characteristics etc. This approach is useful for smaller distributed loads but its usability is limited by the lack of proactive SDA cooperation. Similarly, Pires et al. [42] adopt a Peer-to-Peer (P2P) decentralized model, where every node is both an orchestrator and a computational resource. The system can scale easily by incorporating new volunteer nodes which are rewarded based on a market-oriented approach. However, it does not support cloud integration and only supports edge systems, possibly limiting its scope. Mathias et al. [43] solution describes a hybrid system, where a central aggregator has the overview of all fog nodes while a Fog Orchestrator Agent (FOA), which is an entity executed at each fog node, manages the underlying infrastructure. The twist of this method is that the FOA can act as the local orchestrator in the event of communication loss adding to the adaptability of the system. Nevertheless, the orchestration behavior must be defined beforehand, instead of being based on dynamic policies.

Alam et al. [44] proposed a centralized system designed for publish-subscribe IoT applications, with limited dynamicity, that concentrates basic functions like monitoring and orchestration at the cloud and uses the rest of the network for secondary tasks. Their approach incorporates a data mining component to detect and log anomalous behavior to provide adaptability. On the contrary, Santos et al. [45] adopted a hybrid approach for smart city applications. In their work, they have expanded the ETSI NFV MANO standard to include data monitoring and analysis functionality that is assigned to a central orchestrator running in the cloud. This entity is tasked with additional responsibilities including universal decision-making and fog node management, while the fog orchestrators have a limited scope and manage the local infrastructure. Their framework adopts the OSFP protocol for inter-layer communication and utilizes a GUI to allow users to manually configure the infrastructure's nodes and perform system parameters updates. Another solution designed for smart buildings is IoTEF [46], in which distributed orchestration is combined with a unified management interface that allows the system to be configured at cloud and edge side if the processing power of a node is sufficient. IoTEF's focus is on moving processing closer to the source and minimizes latency and network bandwidth all of which are validated by test results.

[42] A. Pires, J. Simão, and L. Veiga, "Distributed and Decentralized Orchestration of Containers on Edge Clouds," J Grid Comput, vol. 19, no. 3, p. 36, Sep. 2021, doi: 10.1007/s10723-021-09575-x.
[43] M. S. de Brito et al., "A service orchestration architecture for Fog-enabled infrastructures," in 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, May 2017, pp. 127–132. doi: 10.1109/FMEC.2017.7946419.
[44] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah, and Y. Chen, "Orchestration of Microservices for IoT Using Docker and Edge Computing," IEEE Communications Magazine, vol. 56, no. 9, pp. 118–123, Sep. 2018, doi: 10.1109/MCOM.2018.1701233.
[45] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Fog Computing: Enabling the Management and Orchestration of Smart City Applications in 5G Networks," Entropy, vol. 20, no. 1, p. 4, Dec. 2017, doi: 10.3390/e20010004.
[46] A. Javed, J. Robert, K. Heljanko, and K. Främling, "IoTEF: A Federated Edge-Cloud Architecture for Fault-Tolerant IoT Applications," J Grid Comput, vol. 18, no. 1, pp. 57–80, Mar. 2020, doi: 10.1007/s10723-019-09498-8.

3.1.1.1.5 Beyond state of the art

Most of the solutions presented previously compose low level systems that require technical knowledge and expertise to set up and operate. Higher level applications orchestration systems are also essential to implement the full stack of resource orchestration in the edge cloud environment, while obfuscating the complex technical details, e.g., behind an abstraction layer in the form of GUIs or a standardized specification language (e.g. TOSCA), considering issues like application workflows, application deployment, IoT devices, authentication etc. Several solutions of that kind have been provided conceptually in academic papers, with a few of them being developed and deployed as parts of research projects and/or industry initiatives.

Also, the majority of the solutions presented above, employ some sort of central entity with varying degrees of authority and responsibilities over the underlying infrastructure. This centralized architecture is prone to node and connection outages. On the other hand, some of the distributed environments also pose the risk of connection outages and node isolation, offering in practice limited autonomy.

A new distributed architecture is necessary to ensure that distributed resources can operate autonomously without the need of a centralized aggregator. The aim of EMPYREAN is to implement a distributed management fabric, consisting of sets of resources, IoT devices and robots, namely Associations that are managed by Aggregators. Each Association must be able to operate autonomously at a satisfactory level when connectivity to remote cloud resources or other Associations is impossible or undesirable. At the same time, each Association must optimally utilize its heterogeneous resources and achieve inter-Association collaboration, when necessary, through Aggregators' communication to create a collaborative continuum management fabric.

3.1.1.2 Distributed Trust Management

The EMPYREAN project aims to implement a distributed approach to trust management, focusing on the user perspective and Self-Sovereign Identity (SSI), as outlined by C. Allen[47] and supported by initiatives such as the European Self Sovereign Identity framework (eSSIF)[48].

Traditional centralized access control systems, which depend on a trusted third party, face significant drawbacks, including high trust costs and the risk of a single point of failure. Recent studies suggest utilizing decentralized identifiers (DIDs)[49] and verifiable credentials (VCs)[50] as a highly distributed and lightweight alternative to traditional authentication methods. Blockchain technology can replace the registration authority[51] and serve as an authorization

[47] C. Allen, "Decentralized Identity: What's Next?" http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html

[48] European Self Sovereign Identity framework (eSSIF) https://www.eesc.europa.eu/sites/default/files/files/1.panel-_daniel_du_seuil.pdf

[49] DIDs and their applications in decentralized systems https://w3c.github.io/did-core/#did-document

[50] Verifiable Credentials standard https://www.w3.org/TR/vc-data-model

[51] A. Mühle et al., "A blockchain-based approach for the registration authority" https://doi.org/10.1016/j.cose.2018.10.002

mechanism for distributed services, as seen in mobile applications targeting mobile cloud services[52]. For example, smart contracts can dynamically update access permissions and reduce storage overhead by recording users' access privileges on different service providers within a single blockchain transaction.

Further advancements include a proof of concept for a DID registry and a distributed Policy Decision Point (PDP) and Policy Enforcement Point (PEP)[53]. These components facilitate fine-grained, decentralized security policy decisions through languages like Extensible Access Control Markup Language (XACML)[54], allowing a PEP to query PDPs for authorization decisions in a highly distributed manner.

In managing cryptographic keys within the cloud-to-edge continuum, a Distributed Key Management and Authentication (DKMA) solution can address issues that centralized key distribution methods cannot[55]. Cloud-based Key Management Systems (KMSs) such as AWS[56] KMS, Microsoft Azure Key Vault[57], and Google Cloud KMS[58] offer scalable and cost-effective services that can be integrated easily into cloud environments. Additionally, containerization is emerging as a popular deployment model for KMS solutions, providing greater flexibility and portability across various environments.

3.1.1.2.1 Beyond state of the art

The EMPYREAN project advances beyond the current state of the art in distributed trust management by integrating decentralized identifiers (DIDs) and verifiable credentials (VCs) with distributed ledger technologies (DLTs) to create a robust, transparent, and scalable trust management framework. Unlike traditional centralized systems, this approach leverages the immutability and transparency of blockchain technology to ensure verifiable and tamper-proof records of identity and access transactions. Additionally, the use of attribute-based credentials and zero-knowledge proofs (ZKPs) enhances privacy by allowing secure authentication and authorization without disclosing unnecessary personal information. These innovations collectively contribute to a more secure, user-centric, and privacy-preserving trust management system that is capable of meeting the complex requirements of modern distributed computing environments. The integration of these advanced cryptographic techniques and decentralized technologies positions, brings EMPYREAN at the forefront of trust management innovation, providing a comprehensive solution that *addresses the challenges of scalability, security, and privacy in a heterogeneous IoT-edge-cloud continuum.*

[52] Yu L. et al., "Smart contracts for mobile cloud services" https://doi.org/10.3390/s23031264
[53] N. Fotiou et al., "Proof of concept for a DID registry and distributed PDP/PEP" https://doi.org/10.1145/3338507.3358622
[54] G. Peterson, "Decentralized security policy decisions with XACML"
[55] S. Kahvazadeh et al., "Distributed Key Management and Authentication (DKMA) solution" https://doi.org/10.1109/CIoT.2018.8627114
[56] AWS Key Management Service (KMS) https://aws.amazon.com/kms
[57] Microsoft Azure Key Vault https://azure.microsoft.com/en-us/services/key-vault
[58] Google Cloud KMS https://cloud.google.com/kms

3.1.1.3 Secure Shared Data Management

3.1.1.3.1 Blockchain-based Data Sharing

Blockchain technology offers an immutable and decentralized method for sharing and managing data. It has found applications in various fields, including finance, healthcare, and energy sectors[59]. Utilizing a distributed data model based on Blockchain enhances security, boosts performance efficiency, and provides numerous benefits for data management and sharing. These benefits include heightened security, untampered data streams, integrity, and increased trust among stakeholders[60]. The absence of centralized data management ensures data security and system integrity, eliminating the risk of a single point of failure.

Distributed ledger technologies have recently been adopted across a broad range of domains, from data governance to specific sectors like agriculture, healthcare, and transportation [59,60].

Blockchain serves as a digital ledger of transactions, replicated and distributed across the entire network. It provides a decentralized system for managing data and transactions within a peer-to-peer network. Unlike traditional data sharing through Web 2.0 clouds, this decentralized system segments data into blocks, each secured against unauthorized modifications. Any data change must be validated by all peers or miners within the network using advanced cryptographic techniques.

3.1.1.3.2 Beyond state of the art

In the EMPYREAN project, advancements will push the boundaries of current blockchain technology by integrating it into a highly innovative, collaborative IoT-edge-cloud continuum. The project will establish a distributed trust management framework, utilizing blockchain to ensure continuous, dynamic trust assessment and validation. This approach will enhance the security, efficiency, and reliability of data sharing across IoT devices, edge resources, and cloud platforms. By minimizing data reconstruction overheads and improving small message performance, EMPYREAN aims to create a seamless, resilient, and scalable infrastructure for future hyper-distributed applications and services.

3.1.1.3.3 Trustworthy Data Sharing

Trustworthiness, particularly within the context of IoT, such as industrial Internet/Industry 4.0 and autonomous systems, has been a significant topic of discussion. The Industrial Internet Consortium (IIC) has established the foundation of a Trustworthiness model through several papers. This IIC-based model defines Trustworthiness as a combination of System Characteristics, including Security, Reliability, Resilience, Privacy, and Safety, as illustrated by

[59] A. A. Monrat, O. Schelén and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," IEEE Access, vol. 7, p. 117134–117151, 2019.
[60] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," Future generation computer systems, vol. 82, p. 395–411, 2018.

the Trustworthiness Radar[61]. Discussions with the Japanese Robot Revolution Initiative and the German Platform Industry 4.0 have extended these characteristics, incorporating Integrity and considering a selection of other system characteristics, ultimately advancing standardization efforts [62,63].

In parallel, the HORIZON 2020 project SecureIoT formalized the original IIC model, introducing a staggered evaluation of a weighted sum of characteristics, consisting of further weighted sums of attributes and the degree of fulfillment of related properties. Initial metrics for quantification and measurement were outlined in[64]. Trust levels, akin to the security levels described in IEC 62443, illustrate the complexity of specifying Trustworthiness, as it depends on intended use and selected measures. These levels can be seen as maturity levels of the IoT system related to each system characteristic. The project highlighted that a technical evaluation of Trustworthiness requires further research on quantification and measurement standards, policies, and legal descriptions in application areas.

Another research stream considered Trustworthiness in the context of autonomous systems. In the area of autonomous vehicles, non-functional aspects like Availability, Usability, Ethics, Legal Compliance, and Robustness were introduced[65]. These "emerging" characteristics are not implemented within autonomous systems but should be included in a Trustworthiness concept prior to system implementation.

Whether as a basis for continuous evaluation of measurable Trustworthiness or as part of pre-development analysis, the objectives of the characteristics representing policies need alignment, and their interdependencies must be conflict-free. Moreover, the selection of characteristics, attributes, and properties should be done carefully and potentially dynamically.

3.1.1.3.4 Beyond state of the art

The quantification and evaluation of characteristics attributes and properties should incorporate existing trust assurance methods, such as standards or compliance guidelines. This involves specifying appropriate metrics, which might include monitoring device behavior rather than adhering to fixed catalogs. This task relates to T3.1 Distributed Trust Management, Trust Propagation, and Verification, and T3.2 Data Management for Distributed Data

[61] J. Neises and T. Walloschke, “Trustworthiness as Key Enabler for Connected Services in Mobility,” 12 February 2021. [Online]. Available: https://standards.ieee.org/wp-content/uploads/import/documents/other/e2e-presentations/feb-2021/02-Trustworthiness_as_Key_Enabler_Connected_Services_in_Mobility.pdf.

[62] EFFRA Innovation Portal - ISO/IEC 30149 - Internet of things (IoT) -Trustworthiness framework https://portal.effra.eu/result/show/4086

[63] ISO/IEC AWI 30149 Internet of things (IoT) — Trustworthiness framework https://www.iso.org/standard/53269.html

[64] T. W. C. G. J. Neises and B. Popovici, “Trustworthiness as facilitator of Policy and Access Management in Supply Chains,” 12 February 2021. Available:https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/2021-conference-volume-industrie40-security.pdf?__blob=publicationFile&v=5.

[65] H. J. Putzer and E. Wozniak, “Trustworthy Autonomous/Cognitive Systems – A Structured Approach,” Whitepaper, 2020.

Processing. Additionally, mechanisms for resolving non-matching policies must be defined. Metrics, policies, and conflict resolution methods likely depend on business requirements and may vary across different use cases. These challenges will be addressed within the EMPYREAN project.

3.1.1.4 Distributed Data Communication Management

An increasing number of systems span from the data-center down to the microcontroller and need to smoothly operate across this continuum composed of extremely heterogeneous network technologies and computing platforms. Building these systems today is quite challenging due to limitations of the existing technological stacks that are explained below.

Connectivity Islands: Existing communication protocols were designed to work on a very specific use case and in a way address a "connectivity island." As an example, the Data Distribution Service (DDS) [66]was designed to provide a pub/sub protocol that works best for applications running on resourceful hardware connected by multicast-enabled (UDP/IP) wired Local Area Network (LAN). Another assumption in DDS' design is that peer-to- peer communication is quintessential and most of the applications consume data from every other application.

At the opposite side of the spectrum we have MQTT,[67] which was designed to support pub/sub via a client to broker architecture over TCP/IP networks. What is interesting is that both DDS and MQTT provide pub/sub. Yet, their implementations force onto the user very specific communication topologies that are completely orthogonal to the concept of pub/sub. This introduces architectural inflexibility and scalability issues. As an example, DDS is notoriously hard to work with and scale on a Wide Area Network (WAN) as a consequence of its (flat) peer-to-peer only model and its reliance on multicast IP. MQTT, on the other hand, makes communicating across a WAN easy, as far as one can accept to have a hub-and-spoke architecture and a topology not ideal for several edge applications.

But things are even worse. While Message Queueing Telemetry Transport (MQTT) is often referred-to as a lightweight protocol, it relies on TCP/IP and this is not always available nor desirable for constrained hardware and constrained networks. Thus, other protocols are often used to deal with constrained hardware, such as Constrained Application Protocol (CoAP) [68]

At this point, the legitimate question to ask is: how can we deal with systems that include constrained hardware and networks, require high-performance peer-to-peer on the edge, and need to efficiently scale over the Internet? Thus far, the solution has been to use different protocols on different segments of the system and integrate them together hoping to have some meaningful end-to-end semantics. This is tedious, error-prone and inefficient. A

[66] "The data distribution service," 2017. [Online]. Available: http://omg.org/spec/dds
[67] OASIS, "Mq telemetry transport (mqtt) v3.1.1 protocol specification," OASIS, Tech. Rep., October 2014.
[68] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/info/ rfc7252

consequence of the inability of established protocols to deal with the cloud-to-device continuum – they weren't simply designed for it.

Data in Motion and Data at Rest: Pub/Sub protocols have emerged as the technology of choice to deal with data in motion, while databases as the technology of choice to deal with data at rest. These two technology ecosystems are intimately related. Data in movement needs, at some point, to be stored, thus becoming at rest, and eventually retrieved. Yet, from a programmer perspective there is no unified API to deal with both of them. Additionally, while pub/sub features location transparency, databases are location centric. In other words, when expressing a subscription in a pub/sub system one doesn't need to know the location of the publisher(s), yet, when submitting a query it is required to know the location of the database. As a consequence, either one has to keep all the data in a central location – like the solutions provided by cloud storage – or has to deal with the complexity of tracking data's location. This is a major challenge for edge applications. As for these applications it is key to have data stored in a distributed manner, to avoid the cost, including energetic cost, of shipping it to the cloud and to reduce the latency to retrieve it.

Computations: While distributed applications can be modelled as data flows, with computations being triggered only by data, it is rare that a distributed application is entirely based on this paradigm. Often it is convenient to have services and be able to trigger, and invoke their execution. This in turn requires reliance on yet another technology ecosystem that supports request/reply. Which means that in turn the developer needs to learn yet another set of abstractions and APIs. Additionally, existing request/reply frameworks are host-centric, making it hard to deal with load-balancing, and fault-tolerance.

3.1.1.4.1 Beyond state of the art

EMPYREAN's decentralized and distributed data management stack proposes a single solution for data distribution that can work efficiently from the server-grade cloud data-center to the embedded micro-controller and extremely constrained networks. Such data fabric should bridge communications between the enterprise and the embedded world. Additionally, data-centric message protocols enable the user to express interest in a given topic, without any concern about the location of the data source. This characteristic is called location transparency. EMPYREAN's data fabric proposed is agnostic to the underlying technology, implementing a networking layer capable of running above a Data Link, Network, or Transport Layer from the OSI stack. Having a custom routing and networking protocol, enables EMPYREAN to reduce energy consumption by exploiting decentralized and local communication i.e. peer-to-peer, brokered, client-server, without having to send all data through a centralized data-center in the cloud.

3.1.1.5 Secure Edge Data Storage

Secure and highly available data storage solutions are of utmost importance for both businesses and home users. In the context of data storage, security should be linked to the ability of the storage provider to protect data from malicious actors who seek to compromise either its confidentiality or integrity. Meanwhile, availability relates to the user's ability to retrieve and use data when desired. A common scenario when storing data is to use a cloud provider such as Amazon Web Services, Google Cloud or Microsoft Azure. As part of their service portfolio, they support uploading and downloading data with specific guarantees on data storage reliability.

However, using a single cloud location results in three key challenges. First, when it comes to security, users are generally concerned about their data privacy, but tend to make decisions about data sharing and consents based on trust and perceived relation between costs and benefits[69]. This phenomenon could potentially lead users to trust an untrustworthy provider, causing privacy issues. Second, as described in Amazon Web Services (AWS) documentation[70], data availability is often measured as a percentage of uptime over a period of time. At some point, downtime will occur and users' data will become unavailable. Third, long-term data reliability (also referred to as durability) is a common concern for applications that utilize cloud storage. While large-scale permanent data loss events are rare, they are not without precedent. For example, several companies who stored their data solely in OVH's Strasbourg data centers experienced data loss as part of a fire event in 2021[71].

Traditionally, multi-region replication has long been the standard solution to increase data availability and reliability. However, it does not provide additional security benefits and it comes at a significantly greater storage cost compared to using a single cloud location. Furthermore, it does not protect against some of the outages providers experience. Custom solutions that replicate data across different providers are better in this regard, but typically are cost-prohibitive due to the need to move data to and between different clouds. To alleviate this problem, SkyFlok distributes erasure coded versions of user data to multiple cloud providers and locations. Thus, no single provider or location has sufficient information to recover the original contents. In addition, SkyFlok uses a novel erasure coding technique called Random Linear Network Coding (RLNC), which has additional privacy benefits due to the random coefficients used to generate the erasure coded fragments[72]. Compared to using

[69] Andree E. Widjaja, Jengchung Victor Chen, Badri Munir Sukoco, and Quang-An Ha. 2019. Understanding users' willingness to put their personal information on the personal cloud-based storage applications: An empirical study. Computers in Human Behavior 91 (2019), 167–185. https://doi.org/10.1016/j.chb.2018.09.034

[70] Amazon Web Services. 2023. Reliability Pillar - AWS Well-Architected Framework. https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/reliability.html

[71] Law Office of S. Grynwajc, PLLC, 2023, OVH must pay more than 400,000 € after a fire destroyed its data centers – why this decision is important for hosting providers hosting EU personal data?, https://www.transatlantic-lawyer.com/ovh-must-pay-more-than-400000-e-after-a-fire-destroyed-its-data-centers-why-this-decision-is-important-for-hosting-providers-hosting-eu-personal-data/

[72] Alejandro Cohen, Rafael G. L. D'Oliveira, Salman Salamatian, and Muriel Médard, 2021, Network Coding-Based Post-Quantum Cryptography. IEEE Journal on Selected Areas in Information Theory 2, 1 (2021), 49–64. https://doi.org/10.1109/JSAIT.2021.3054598

a single cloud, multi-cloud solutions force malicious actors to compromise the security of multiple providers and locations, a significantly greater challenge.

3.1.1.5.1 Beyond the state of art

To improve data access latency and further reduce the need to trust cloud providers, SkyFlok has been extended in the SERRANO project [73] to support edge storage locations. By supporting edge locations, prospective users are able to choose between different storage policies. They can store data either in the cloud, on the edge, or using both through a hybrid approach. In all cases, they can specify the exact locations as well as the desired level of reliability. This feature set that advances the state of the art has been proven to be effective through evaluation performed on a prototype implementation. In EMPYREAN, CC will continue to increase the maturity of this solution.

Traditional storage solutions have relied on a system of tiers to provide users with different choices to trade off availability, reliability, performance, storage and access costs. Through the use of customizable erasure coding, and support for edge-cloud hybrid storage, we do away with the discrete tier system. Instead, users are provided with a seamless, tierless storage continuum. By greatly increasing the granularity of decisions with which the aforementioned trade-offs can be approached, flexibility is gained which translates into a system that is better able to meet users' requirements.

The current extension of SkyFlok goes beyond what competitors offer through the aforementioned features. These features are achieved through the careful coordination of the SkyFlok backend. This is a highly redundant cloud-hosted system. Beyond the many benefits, it unfortunately comes with some limitations. Should the cloud become unavailable or the user's internet connection fail, data cannot be reached even if it is stored exclusively at the edge. To get around this issue, we propose a further extension to SkyFlok which allows temporary autonomous operation at the edge. To further increase user confidence, we plan to offer users the ability to supply and store encryption keys locally. While compromising reliability to some degree and placing some security burdens on the users' shoulder, the benefits are clear. Commercial cloud storage solutions such as Tresorit[74] offer comparable features.

3.1.1.6 RDMA for Edge Nodes Supporting IoT-based Sensor Data

Remote Direct Memory Access provides a lower CPU codepath length for network I/O that provides significant performance gains. This is achieved by offloading the network Transport layer to an accelerator onto the NIC and allowing the NIC peripheral to directly read/write into the application address space buffers. In the described context, the CPU merely orchestrates the network I/O transfers, by simply initiating transfer requests and checking for their completions. Notably, the most popular RDMA API (named Verbs) allows developers to implement in a straightforward manner the Proactor Software model where the request path

[73] https://ict-serrano.eu/

[74] "Tresorit: Encrypted Cloud Storage for Businesses." Available: https://tresorit.com/

is totally decoupled from the completion path, which enables the application to generate I/O requests in a pipelined manner. The approach provides significant performance gains, especially for small-sized I/O accesses because it favors request and completion aggregation and their handling in batches, at all levels of the network stack.

Given the described RDMA characteristics, it is a good fit to act as the backhaul network fabric that connects an edge node to other edge nodes or centralized cloud compute resources. An IoT edge node may collect a storm of small messages, which can be transported towards centralized compute resources with less jitter and as they are captured effectively forming a distributed pipeline with the compute nodes.

RDMA I/O is popular in HPC applications and typically underpins AI training on large clusters where latency and bandwidth performance are of paramount importance. Since RDMA relieves the CPU from network I/O tasks it can provide a significant boost on resource constraint devices, which can dedicate more compute resources to actual data manipulating tasks rather than having to deal with data transfer tasks as well. For a number of reasons, which are primarily challenging programmability and NIC hardware dependency, leveraging RDMA in IoT edge node deployments is currently largely unexplored.

3.1.1.6.1 Beyond the state of art

While RDMA performance gains is critical for efficient edge node operation going forward, the RDMA verbs API is very low-level and difficult to integrate with IoT pipelines or application specific appliances (e.g. FPGA-based edge devices). For the former case the software-based integration of RDMA requires an interface that is typically used for communication in IoT like pub/sub communication. In the latter case a hardware interface is required that provides the right streaming interface abstraction that can introduce network I/O as a pipeline extension to the existing hardware operation. At EMPYREAN we introduce the flexdriver framework for RDMA that provides the right interface abstractions to integrate RDMA operation in IoT edge nodes be it hardware or software frameworks.

The goal is to leverage RDMA and accelerate existing communication frameworks that are typically used in IoT deployments like pub/sub brokers, by changing/extending appropriately the internals, while using the same API that allows for easy integration with existing IoT applications.

3.1.1.7 Cross-platform and Lightweight Container Packaging

The convergence of IoT, edge, and cloud infrastructures necessitates efficient mechanisms for workload execution, particularly for data analytics and AI applications. This drives the need for lightweight containerization solutions that maximize performance and minimize overhead. Unikernels have emerged as a compelling option due to their minimalist philosophy. Unlike traditional operating systems, unikernels compile applications directly into single-purpose images with only the necessary resources, which enhances security and performance efficiency. Notable examples include MirageOS and OSv, which have demonstrated

substantial improvements in boot times and memory usage. This article[75] provides empirical evidence supporting their benefits in cloud-native environments. Unikraft is an advanced, flexible unikernel development framework designed to simplify and optimize the process of creating unikernels. Developed under the auspices of the Xen Project, Unikraft leverages a component-based architecture, which allows developers to selectively include only the minimal set of libraries and runtime components that their application necessitates. This modular design ensures that unikernels built with Unikraft are highly optimized both in terms of resource usage and performance, which is particularly advantageous for IoT, edge, and cloud environments. Unikraft distinguishes itself through several key innovations. Firstly, its configuration and build process is highly automated, significantly reducing the complexity associated with creating and maintaining unikernels. Unikraft supports a diversity of platforms and architectures, including x86, ARM, and various hypervisors, thereby facilitating broad applicability across different hardware and virtualization scenarios. A study by Kuenzer et al.[76], presents benchmarking results showcasing Unikraft-built unikernels' superior performance metrics compared to traditional VMs and even some other unikernel solutions. The modular nature of Unikraft allows for reductions in boot times and memory footprints, while still providing competitive execution speeds—characteristics that are essential for latency-sensitive applications in edge computing and resource-constrained IoT devices.

Furthermore, Unikraft's ecosystem supports popular programming languages and runtime environments, such as C, C++, Python, and Go, enhancing its usability for a wide range of applications from microservices to complex AI workloads. This support is crucial for developers needing to port existing applications to unikernel environments with minimal re-engineering efforts, thus expediting deployment and scalability in heterogeneous IoT-edge-cloud ecosystems.

Open Container Initiative (OCI) containers prioritize standardization and compatibility across various platforms. OCI containers, such as those implemented through Docker and Kubernetes, encapsulate applications along with their dependencies. This guarantees that a given container will run consistently regardless of the underlying environment. This approach not only enhances cross-platform portability but also facilitates orchestration and management at scale, which is particularly beneficial for managing distributed AI workloads and data analytics tasks across hybrid IoT-edge-cloud ecosystems. Felter et al.[77], illustrate how these containers can efficiently handle intensive computational tasks, making them a pragmatic choice for modern data workflows.

[75] Manco, V., Arnal, G., Brandan, C., Macialek, D., Martin, B., & Watson, I. A. (2018). Unikernels: Library Operating Systems for the Cloud. ACM Computing Surveys (CSUR), 51(6), 134.

[76] Kuenzer, S., Wojcik, M., Girol, T., Pacher, M., Lankes, S., Beri, P., & Fritsch, S. (2021). Unikraft: Fast, Specialized Unikernels the Easy Way. ASPLOS '21: Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems.

[77] Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015). An updated performance comparison of virtual machines and Linux containers. 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA.

Complementing these technologies, the NIX functional package manager offers a declarative approach to software deployment, enabling reproducibility and consistency in hybrid infrastructures. NIX's purely functional environment ensures that system dependencies are isolated, reducing the risk of "dependency hell" and enabling seamless transitions between development and production environments. This is particularly relevant for AI applications that often demand specific library versions and configurations. Research, such as this article[78] by Dolstra et al., provides a comprehensive overview of how NIX can streamline software management and ensure that applications deployed across different stages of the IoT-edge-cloud continuum maintain their integrity and performance. This functional paradigm aligns well with the modular and microservices architectures prevalent in modern IoT and edge computing frameworks.

3.1.1.7.1 Beyond state of the art

In the context of EMPYREAN, we are going to propose techniques making use of NIX functional package manager to efficiently build unikernel environments based on OCI standards. This will enable a functional, reproducible way to build OCI-based unikernel images. In our studies we will also consider applications needing to be executed on accelerators but since unikernels do not have this ability yet we need to consider techniques enabling the separation in different images.

3.1.2 Decentralized Intelligence and AI-enabled Application Development and Deployment

3.1.2.1 IoT-Edge-Cloud resource orchestration

IoT-edge-cloud resource orchestration algorithms are a vibrant research area, relying on multi-objective optimization[79,80] and encompassing a diverse set of techniques, including heuristic[81], metaheuristic[82], and machine learning[83] based algorithms that aim at handling the increased complexity of resource orchestration across multi-domain heterogeneous environments. Additional challenges include: the infrastructure size, its dynamicity, scalability and energy efficiency issues, along with other conflicting optimization criteria that in many

[78] Dolstra, E., Löh, A., & Pierron, O. (2010). NixOS: A Purely Functional Linux Distribution. Journal of Functional Programming, 20(5-6), 577-615.
[79] H. Yuan, "Energy and performance-optimized scheduling of tasks in distributed cloud and edge computing systems," 2020.
[80] Luc Angelelli, Anderson Andrei Da Silva, Yiannis Georgiou, Michael Mercier and Gregory Mounié Towards a Multi-objective Scheduling Policy for Serverless-based Edge-Cloud Continuum. CCGrid2023, May 2023
[81] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "Optimized placement of scalable iot services in edge computing," in 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019, pp. 189–197.
[82] S. Wang, Y. Li, S. Pang, Q. Lu, S. Wang, and J. Zhao, "A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline," Sci. Program., vol. 2020, p. e3967847, Mar. 2020, doi: 10.1155/2020/3967847.
[83] Zheng, T., Wan, J., Zhang, J. et al. Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing. J Cloud Comp 11, 3 (2022).

cases lead to a very large solution space. Several related surveys are available in this area. Asuncao et al. [84] studied resource management challenges regarding hybrid deployments including IoT and Edge. They consider that managing task scheduling and allocation of heterogeneous resources along with adapting an application to current resource and network conditions will require the development of new schedulers and that allocations must be dynamic enough to support migration.

3.1.2.1.1 Typical Methodologies

The most prevalent methods to tackle the resource scheduling problem have been categorized as static approaches, including heuristics, mathematical models (Euclidean formulation), mathematical optimization-based methods such as Integer Linear Programming (ILP), Mixed Linear Programming (MLP), etc. Although exact techniques such as Branch and Bound can obtain the optimal solutions to ILP and MILP formulated problems, they are not well-suited for the extensive search space of large-scale edge-cloud infrastructures and heavy IoT workloads. Hence, heuristic algorithms perform comparatively better at finding near-optimal solutions in minimal time.

The authors in [85] aim to maximize the energy efficiency of IoT devices, by offloading computations to the edge-cloud hierarchy. Utilizing a simple yet effective multi-criteria decision-making technique named TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution), they aim to maximize a weighted combination of the energy and time savings offered by task-offloading. At each step, the mechanism selects the computing node which is the closest to the positive ideal solution and furthest from the negative ideal solution, according to the specified weighted objective. In [86], a QoE-aware placement policy for IoT applications in fog environments is proposed. Employing two fuzzy logic models, the authors evaluate application expectations and fog node performance and feed a linear optimization mechanism to perform the placement.

Authors in [87] delves into the problem of microservice placement in the edge-cloud continuum, utilizing network devices along with standard computing nodes for task processing, in a concept known as “in-network computing”. The objective is the minimization of the total cost, encompassing communication, operational and deployment expenses. The study also considers potential node-failures within the infrastructure. An ILP is presented and subsequently proven NP-hard by reduction to the Generalized Assignment Problem (GAP). A best-fit heuristic is developed to tackle the prolonged execution times for more complex

[84] M. D. de Assuncao, A. da Silva Veith, and R. Buyya, “Distributed data stream processing and edge computing: A survey on resource elasticity and future directions,” J. Netw. Comput. Appl., vol. 103, pp. 1–17, 2018.
[85] G. Castellano, F. Esposito, and F. Risso, “A Service-Defined Approach for Orchestration of Heterogeneous Applications in Cloud/Edge Platforms,” IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1404–1418, Dec. 2019, doi: 10.1109/TNSM.2019.2941639.
[86] A. Pires, J. Simão, and L. Veiga, “Distributed and Decentralized Orchestration of Containers on Edge Clouds,” J Grid Comput, vol. 19, no. 3, p. 36, Sep. 2021, doi: 10.1007/s10723-021-09575-x.
[87] M. S. de Brito et al., “A service orchestration architecture for Fog-enabled infrastructures,” in 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, May 2017, pp. 127–132. doi: 10.1109/FMEC.2017.7946419.

configurations. Li et al. [88] proposed a hybrid computing system for smart factories and Industry 4.0 by proposing a four-level architecture that integrates the historical heritage of computational resources. Furthermore, a two-phase resource scheduling strategy is introduced; the selection of edge computing servers is done by taking into consideration different factors corresponding to low real-time constraints in phase 1, whereas phase 2 manages cooperation amongst multiple edge servers in order to construct an Edge Server Cluster (ESC). To tackle the excessive network traffic generated by IoT ecosystems, the work in [89] proposes a dynamic master-slave SDN controller scheme, with a load-balancing approach to intelligently allocate network resources and mitigate congestions and failures. A sophisticated mechanism ranks the slave controllers based on a weighted combination of key metrics such as their current CPU and memory utilization. Based on the mechanism's decisions, the switches from the IoT network are dynamically assigned to the selected slave controllers. The migration of the switches between controllers is modelled as a binary knapsack problem, where slave controllers represent containers, while switches in local IoT networks are regarded as the objects.

3.1.2.1.2 AI-based solutions

ML-based techniques utilized in the context of optimizing resource allocation include k-means clustering, Decision Tree Regression (DTR), Multiple Linear Regression and Naïve Bayes. However, the latest works are using Reinforcement Learning (RL)-based techniques due to the need to dynamically adapt to the ever-changing IoT-based scenarios. The main benefit lies in the fact that RL-based techniques such as Q-Learning and State-Action-Reward-State Action (SARSA) do not require any dataset for training the model. Furthermore, such techniques operate iteratively with the goal of maximizing rewards for the agent, which takes actions based on the environmental state. The drawbacks such as the inability to handle high-dimensional state information regarding incoming IoT tasks, can be resolved by utilizing techniques that include Deep Reinforcement Learning (DRL), Deep Q-Learning (DQN) and Deep Neural Network (DNN).

Authors in [90] examine the problem of edge-cloud cooperative content-delivery in asymmetrical Internet of Vehicles (IoV) environments, with the aim of minimizing total network delay by providing optimal computing, caching and communication resource allocation. To fetch a targeted file, the autonomous vehicle first communicates its request with the access Base Station (BS). If the content is cached in the BS, it immediately replies, otherwise the request is forwarded to adjacent BSs and the cloud. The delay that includes waiting and serving latencies at a node, is modeled based on queueing theory, and specifically

[88] E. Al-Masri et al., "Energy-Efficient Cooperative Resource Allocation and Task Scheduling for Internet of Things Environments", Internet of Things, vol. 23, pp. 100832–100832, 2023, doi: https://doi.org/10.1016/j.iot.2023.100832.

[89] R. Mahmud, S. N. Srirama, K. Ramamohanarao, R. Buyya, "Quality of experience (qoe)-aware placement of applications in fog computing environments", Journal of Parallel and Distributed Computing, vol. 132, pp 190–203, 2019, doi: https://doi.org/10.1016/j.jpdc.2018.03.00

[90] Ali, S.O. et al., "CaMP-Inc: Components-aware microservices placement for in-network computing cloud-edge continuum", IEEE Global Communications Conference (GLOBECOM), 2022, doi:10.1109/globecom48099.2022.10000936.

the M/M/k queuing system. Finally, a DQN-enabled cross-layer collaborative caching and routing scheme is developed to tackle the optimization problem. The reinforcement learning mechanism makes collaborative routing and caching decisions by predicting content popularity, based on the request's history information and current network status.

Authors in [91] examine the dynamic task offloading from IoT devices to edge-cloud continuum, aiming to minimize a weighted combination of delay and energy consumption. QoS constraints include the successful decoding probability from the end-devices. A hybrid DRL-based framework is proposed based on centralized training- decentralized execution. Each IoT device simulates an actor while all devices share a common critic. A double DQN network is introduced to address the over-estimation problem of standard DQN, while accelerating training convergence. [92] investigates the problem of ML task offloading (particularly Convolutional Neural Network - CNNs) from IoT devices to nearby cloudlets in order to minimize a weighted combination of system delay and energy consumption of the end-devices. The problem is decomposed into two separate sub-problems, namely the offloading decision problem, and the resource allocation problem. The first one is tackled utilizing an enhanced DRL algorithm, where the neural network is pre-trained, utilizing the results of an exact Mixed Integer Non-Linear Programming (MINLP) solver. The resource allocation problem is addressed by a Salp Swarm metaheuristic algorithm. Notably, the second algorithm is also used to evaluate the offloading decisions of the DRL agents, in order to improve their policy.

3.1.2.1.3 Federated Learning

AI and ML techniques consolidate the results for better predictions and enhance the user experience. Nevertheless, data is stored at a centralized location, and the models rely on this data for the training. As a result, these methods encounters various challenges including privacy concerns, data security and regulatory compliance. The solution to this problem lies in training the model on the device itself instead of in a centralized server. This is where Federated Learning (FL) comes into play, providing hyper-personalized space, low cloud infrastructure overhead, and prominent privacy preservation while minimizing latency. FL can be treated as a decentralized form of machine learning, which creates a shared model in place of a central data model. The new models are being trained collaboratively on the edge, where the data never leaves the personalized device. Although the devices and machines train several models at distributed locations in parallel and send their collaborative results to a centralized server to create a machine learning model. Therefore, FL leverages both the distribution of data and computational resources while safeguarding data privacy and integrity.

[91] H. Hu, D. Wu, F. Zhou, X. Zhu, Rose Qingyang Hu, and H. Zhu, "Intelligent Resource Allocation for Edge-Cloud Collaborative Networks: A Hybrid DDPG-D3QN Approach", IEEE Transactions on Vehicular Technology, vol. 72, no. 8, pp. 10696–10709, Aug. 2023, doi: https://doi.org/10.1109/tvt.2023.3253905.

[92] Z. Aghapour, S. Sharifian, and H. Taheri, "Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments", Computer Networks, vol. 223, p. 109577, Mar. 2023, doi: https://doi.org/10.1016/j.comnet.2023.109577.

Authors in [93] discuss emerging challenges for supporting intelligent applications in 6G networks. First, key aspects of the upcoming 6G era are presented, such as the inclusion of multi-dimensional resources, including Unmanned Aerial Vehicles (UAVs) and hardware accelerators. Federated learning is introduced as the primary solution to deal with end-user data privacy. A Policy Network Reinforcement Learning (PNRL) algorithm is developed to tackle the multi-dimensional resource allocation problem, aiming to maximize the operator's/provider's revenue. Finally, the authors present existing and imminent challenges for future investigation from the perspectives of model architecture design, model training and inference. Techniques such as network pruning and transfer learning can reduce the execution delay, while model partitioning can distribute the model's execution across the network. Finally, collaborative model training, such as FL, can provide a valuable solution in decentralized environments; however, users should be incentivized to contribute to the model's training, highlighting an important open research area. A virtual network embedding strategy based on horizontal federated learning (HFL) is presented in [94]. The ISP requests a virtual network with given computing/networking characteristics, which are mapped to the underlying physical networks. Federated learning is utilized to ensure data privacy over multi-domain physical networks. The authors examine the dynamic problem, where the characteristics of the underlying physical network vary with time, both resource and topology-wise.

3.1.2.1.4 Swarm intelligence and genetic algorithms

Often inspired by the complex mechanisms of certain animal or flora species found in nature, metaheuristics including swarm intelligence and evolutionary algorithms are preferred in place of traditional heuristics due to their capability to thoroughly explore a vast solution space, effectively avoiding local optima. Yuan in [95] does a thorough study of task scheduling in hybrid edge-cloud environments and proposes various algorithms optimizing energy, performance and cost while considering various constraints such deadlines and QoS.

Maia et al. in [96] considered the offline placement problem of IoT services supporting horizontal and vertical scaling in a hybrid edge-cloud environment. They tried to solve the joint problem of service placement and load distribution to minimize the delay in execution (QoS). They formulated the problem with Mixed-Integer Linear Programming (MILP) method which has high computational complexity, so they considered solutions either through MILP approaches for optimal solution or greedy and genetic methods which provide good placement with less

[93] X. Zhang, P. Han, C. Feng, T. Ma, and L. Guo, "Multi-dimensional Resource Orchestration Toward Edge Intelligence in 6G Networks", IEEE Communications Magazine, vol. 61, no. 12, pp. 46–52, Dec. 2023, doi: https://doi.org/10.1109/mcom.005.2200905.

[94] P. Zhang, et al., "Multi-Domain Virtual Network Embedding Algorithm based on Horizontal Federated Learning", IEEE Transactions on Information Forensics and Security, 2023 , doi: https://doi.org/10.48550/arxiv.2205.14665.

[95] H. Yuan, "Energy and performance-optimized scheduling of tasks in distributed cloud and edge computing systems," 2020.

[96] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "Optimized placement of scalable iot services in edge computing," in 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019, pp. 189–197.

computational complexity and faster results. In our case we would like to consider both offline and online placement optimizations.

Wang et al. [97] in proposed another genetic-based scheduling algorithm under a deadline constraint to enable the execution of tasks upon edge or cloud resources with a goal to minimize the execution time of all tasks. They have enhanced the genetic algorithm with a catastrophic variant to increase the mutation probability to stay away from the current optimal, which may not be the ideal optimal solution, and allow a re-optimization in a later stage. They have provided experimentations and performance evaluations of their algorithms upon the CloudSim simulator.

The authors in [98] investigate the problem of placing multi-service applications to computing nodes, subject to capacity and QoS constraints, such as deadline times of the applications' services. The authors provide a concise yet informative introduction to quantum computing and then proceed to detail their quantum-inspired particle swarm optimization algorithm. Similarly to the standard Particle Swarm Optimization (PSO), the qubits change their quantum state (described by their α and β parameters) based on their personal best and the global best particle. A new solution can be generated by observing the state of the qubits. The effectiveness of the technique lies on the exponential representative power of the qubits, where only a small swarm of N qubits can represent 2^N possible solutions. Guerrero et al. [99] studied and compared three evolutionary algorithms namely, Weighted Sum GA (WSGA), multi-objective evolutionary algorithm based on decomposition (MOEA/D), and NSGA-II to address the problem of service placement in a fog environment. The algorithms were evaluated on the basis of resource usage, service spread, network latency, and execution time. Djemai et al. [100] proposed a hierarchical three-layered fog infrastructure. An energy-efficient IoT service placement strategy, using Discrete PSO (DPSO) is employed to address the service placement problem. The DPSO strategy was evaluated on the basis of energy consumption and delay. The DPSO was compared to other strategies such as Binary PSO (BPSO), and Cloud-Only, and with different layer-interplay combination approaches. The DPSO shows a better trade-off between cloud, IoT, and fog layer usage. A novel metaheuristic algorithm, termed the "whale optimization algorithm" (WOA), is presented in [101]. The problem under consideration is the assignment of edge gateways to end-devices, with a dual objective of

[97] S. Wang, Y. Li, S. Pang, Q. Lu, S. Wang, and J. Zhao, "A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline," Sci. Program., vol. 2020, p. e3967847, Mar. 2020, doi: 10.1155/2020/3967847.

[98] M. Bey, P. Kuila, B. B. Naik, and S. Ghosh, "Quantum-inspired particle swarm optimization for efficient IoT service placement in edge computing systems", Expert Systems with Applications, vol. 236, p. 121270, Feb. 2024, doi: https://doi.org/10.1016/j.eswa.2023.121270.

[99] C. Guerrero, I. Lera, and C. Juiz, "Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures", Future Generation Computer Systems, vol. 97, pp. 131–144, Aug. 2019, doi: https://doi.org/10.1016/j.future.2019.02.056.

[100] Tanissia Djemai, P. Stolf, T. Monteil, and J.-M. Pierson, "A Discrete Particle Swarm Optimization Approach for Energy-Efficient IoT Services Placement Over Fog Infrastructures", IEEE International Symposium on Parallel and Distributed Computing (ISPDC), 2019, doi: https://doi.org/10.1109/ispdc.2019.00020.

[101] A. K. Sangaiah, A. A. R. Hosseinabadi, M. B. Shareh, S. Y. Bozorgi Rad, A. Zolfagharian, and N. Chilamkurti, "IoT Resource Allocation and Optimization Based on Heuristic Algorithm", Sensors, vol. 20, no. 2, p. 539, 2020, doi: https://doi.org/10.3390/s20020539.

minimizing the total communication cost while securing a predetermined load balancing level. Initial solutions (whales) can be produced by any heuristic algorithm. Then, based on an exploration/exploitation trade-off, each whale undertakes an action (e.g., directly moves or spirals towards the best whale, performs random movement etc.) in order to effectively inspect the solution space.

3.1.2.1.5 Game theoretic approaches

Game theory is a formal framework that includes a set of mathematical tools for studying complex interactions between interdependent, rational and self-interested players. Strategic games have a variety of applications in economics, politics, sociology, and other fields. Over the last decade, there has been an increase in studies using game theory to model and evaluate modern communication networks, as well as emerging technologies such as edge-cloud computing and other internet computation platforms.

Authors in [102] provide a game-theoretic approach for resource sharing between coalition partners (e.g., ISPs) for serving edge computing demands. When a server becomes overloaded, it can use the resources of nearby servers to execute its incoming traffic. The optimization objective is to maximize the ISPs' objectives (utility functions), which can vary resulting in a multi-objective optimization problem. A Nash Bargaining Solution (NBS) approach is presented, followed by an iterative distributed algorithm to reach the equilibrium. Furthermore, the fairness of the proposed solution is proven experimentally by calculating the Jain's index. Jiao et al. [103] propose an auction-based resource allocation scheme for edge service providers that provide resources for blockchains. The proposed mechanism maximizes social welfare and guarantees truthfulness, computational efficiency and individual rationality. In a study by Munir et al. [104], a hierarchical game theory algorithm has been proposed for optimal resource allocation in the process of a heterogeneous network with femtocell movements in the edge. The first game of the algorithm consists of Femtocell Access Points (FAPs), which play a non-cooperative game to select between open and closed access policies to increase their home subscriber rates. In the second game, Macrocell User Equipment (MUE) is provided to decide on the connection between the FAPs and the Macrocell Base Station (MBS) to maximize their rates and total network performance. Chen et al. [105] have proposed a framework based on the game theory, named a multi-leader multi-follower Stackelberg game, where End Users (EUs) and Mobile Edge Clouds (MECs) or mobile edge computing nodes perform as followers and leaders, respectively. The proposed framework is a solution for computing a Stackelberg equilibrium, where each MEC reaches a maximum revenue, and each end user attains maximum efficiency with resource budget

[102] Faheem Zafari, P. Basu, K. K. Leung, J. Li, A. Swami, and D. Towsley, "Resource Sharing in the Edge: A Distributed Bargaining-Theoretic Approach", IEEE Transactions on network and service management, vol. 20, no. 4, pp. 4369–4382, 2023, doi: https://doi.org/10.1109/tnsm.2023.3265813.

[103] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social Welfare Maximization Auction in Edge Computing Resource Allocation for Mobile Blockchain", IEEE International Conference on Communications (ICC), 2018.

[104] H. Munir, S. A. Hassan, H. Pervaiz, Q. Ni, "A game theoretical network-assisted user-centric design for resource allocation in 5G heterogeneous networks", IEEE 83rd vehicular technology conference (VTC Spring), 2016.

[105] Y. Chen, Z. Li, B. Yang, K. Nai, K. Li, "A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing", Future Generation Computing Systems, vol. 108, pp. 273–287, 2020.

limitations. [106] profiles users and edge servers as tuples encompassing location, resources, and bandwidth. The edge server first allocates resources in four distinct areas where users are clustered, proportional to the users demands in each location. Then, a Dutch auction is performed to deduce the actual resources and bandwidth to be allocated for each end-user. Finally, the work in [107] targets the dynamic resource sharing in edge clouds, proposing an online incentive mechanism. An estimation mechanism was used to remove the uncertainty of resource requirements by microservices and then an online action framework was utilized to achieve microservice cooperation. In this framework, the proposed winner selection algorithm was proven to guarantee truthful bidding and individual rationality.

3.1.2.1.6 Beyond state of the art

The existing literature primarily focuses on the performance of the developed heuristics, highlighting their sophisticated abilities to handle complex optimization functions and achieve quality system-wide solutions. However, the necessity of a central entity to handle the involved computations is paramount, and little effort has been devoted into distributing the resource allocation process. In a decentralized, dynamic, and highly heterogeneous ecosystem, EMPYREAN will strive not only to globally optimize the infrastructure's performance, but also to detail a completely decentralized execution framework, effectively addressing processing, communication, privacy and conflicting interests between the involved participants.

Also, EMPYREAN will push the state of the art in IoT-edge-cloud infrastructure resource orchestration by developing advanced multi-agent optimization algorithms that leverage game theory with auction-based mechanisms, multi-agent Deep Reinforcement Learning, and swarm intelligence. Through these approaches, we aim to create highly effective mechanisms for speculative resource allocation, allowing for the timely operation in high complexity multi-technology and multi-domain infrastructures. To better address the multiple and conflicting objectives within and among different associations, EMPYREAN will support multiple levels of co-operation (partial/no co-operation). However, as uncertainties are introduced and the number of agents may change (depending on the number of Associations, their objectives, etc.), EMPYREAN will expand machine reasoning and game theoretical foundations of synthesis to design a high-level orchestrator that fulfills the application requirements and guarantees its correct behavior according to the specifications. Also, EMPYREAM, will focus on scalability and design algorithms able to handle multiple autonomous large-scale environments, using distributed and parallel computing techniques for adaptive and robust resource management.

[106] S. Ju, J. Qiu, and W. Song, "Dynamic Resource Allocation Strategy Based on Dutch Auction", IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), 2022, doi: https://doi.org/10.1109/ipec54454.2022.9777410.

[107] A. Samanta, L. Jiao, M. Muhlhauser, and L. Wang, "Incentivizing Microservices for Online Resource Sharing in Edge Clouds", International Conference on Distributed Computing Systems (ICDCS), 2019, doi: https://doi.org/10.1109/icdcs.2019.00049.

3.1.2.2 AI-based Workload Autoscaling

AI-based workload autoscaling has attracted a lot of interest by the community since it is an area that can enable important optimizations. These are the main approaches followed in the different research articles.

3.1.2.2.1 Proactive forecast and adjustment

This approach contains two steps: forecasting future resource usage using ML algorithm, and adjusting the container limit based on the forecasted usage. A small number of articles use this approach.

Jeong B et al[108] proposed a technique, which used Long short-term memory (LSTM) network as the resource predictor. The heuristic rule is an algorithm taking the previous request and the predicted resource usage (by LSTM) as input, and output the next container limit.

Luong et al[109] used Linear Regression and Auto Regressive Integrated Moving Average algorithm to predict the resource usage in the near future, and uses FFT integrated Welch filter and periodograms techniques to forecast the long-term future. These predictions are used to scale resources using a threshold comparison method.

General advantages of this category:

- Uncoupled models (ML, Rule part), facilitate development and maintenance.
- Offline training, avoids poor performance during the startup phase.

General drawbacks of this category:

- Weak adaptability to different/dynamic cloud environments.
- Partly existing heuristic rules, need expertise to customize.

3.1.2.2.2 Reinforcement learning

A reinforcement learning algorithm can be used to directly forecast the next container limit. In RL algorithm, the autoscaler acts as RL agent, to detect environment (historical usage) and make decisions (container limit). A large number of articles using this approach, with different RL technologies. Most of them use deep reinforcement learning models because of the large continuous state space in scaling problems. The model can be pre-trained offline or be trained online.

This article[110] raises a deep Q learning model with autoencoder and weight sharing structure to dynamically allocate VM resources to the cloud server, aiming at reducing energy

[108] Jeong, B. *et al.* (2023) 'Stable and efficient resource management using deep neural network on cloud computing'.
[109] Luong, D.-H. *et al.* (2018) 'Predictive Autoscaling Orchestration for Cloud-native Telecom Microservices'.
[110] Liu, N. et al. (2017) 'A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning'.

consumption. Offline data is used to pre-train this model to derive a first version of the correlation between Q values and state-action pairs.

This publication[111] performs vertical scaling on CPU, memory, I/O and network for each component in microservices using a novel RL algorithm called Deep Deterministic Policy Gradient, based on their historical usage and the critical paths extracted from the microservices relation graph. No offline pre-training is applied in this model.

General advantages of this category:

- Strong adaptability to different/dynamic cloud environment.
- Highly transparent automatic policy, independent on human intervention.

General drawbacks of this category:

- Long online training time at the beginning, with poor performance. Can be partially mitigated but will reduce adaptability.
- Hard to develop for stable best performance: coupled complex interactive system, hyperparameters and randomness greatly matters.

3.1.2.2.3 Hierarchical models

In this approach, different sub-models exist in the autoscaler to read the historical usage and give predictions on container limit. A global chooser algorithm is responsible for dynamically picking the best sub-model based on the sub-models' performance. The representative work is Google Autopilot[112]. This publication uses simple "argmin" functions as sub-models. These sub-models take CPU and memory usage histograms as input, and output corresponding limit prediction every 5 minutes. A loss function considering overrun and underrun costs, and model-switching costs is used as the global chooser, to pick the best sub-model dynamically. The advantages of this work is its good interpretability and simple implementation.

Toka et al.[113] use different machine-learning algorithms as submodels to predict future limit (e.g. LSTM, AR, HTM and RL). A global evaluator calculates the recent accuracy of their predictions, and chooses the best machine-learning algorithm dynamically.

General drawbacks of this category:

- Both are reactive approaches, not proactive.
- Changing the sub-models dynamically may cause more SLOs.

3.1.2.2.4 Beyond state of the art

[111] Qiu, H. et al. (2020) 'FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices'.

[112] Rzadca, K. et al. (2020) 'Autopilot: workload autoscaling at Google'.

[113] Toka, L. et al. (2021) 'Machine Learning-Based Scaling Management for Kubernetes Edge Clusters'.

In the context of EMPYREAN, we will study, implement and execute representative algorithms from each category, and compare them with continuum-native representative workloads. The goal is to find a generally dominating performance on resource utilization and different metrics, compared to the rule-based approaches. In particular, we will adopt the work on Autopilot considering CPU and RAM limits adjustment and we will consider in particular techniques to minimize the Out-Of-Memory errors while improving bin-packing in the context of workload autoscaling. Possible variations and enhancements of the ML-algorithms will be also considered by sufficiently adapting the hyper-parameters of the ML techniques to better fit the needs of our context and workloads.

3.1.2.3 Workflow-based Continuum-native Application Design

The Workflow Management system is responsible for the automation of orchestration and execution of task collections upon computational resources. A common pattern in scientific and cloud computing involves the execution of many computational and data manipulation tasks which are usually coupled i.e., output of one task used as input on another. Hence coordination is required to satisfy data dependencies. The workload of task execution is divided among the available distributed computational resources. Consequently, this introduces further complexity related to processes such as load balancing, data storage, data transfer, tasks monitoring and fault-tolerance. Automation of the aforementioned aspects of the orchestration process has led the creation of workflow management systems.

A study by Deelman et al. [114] related to scientific workflow management analyzes the current state of the art on these systems and provides future research challenges. On the HPC side there are some particular tools such as Taverna[115], Pegasus[116] and Makeflow[117] that have been used in production since years with different maturity levels. Those systems generally share many similarities in their concepts. They all have one or more principal languages to program workflows and they provide connections towards specific resource management systems for the deployment part. They have been designed with scalability and fault tolerance in mind and their multiple years in production has allowed them to make a lot of progress on the interoperability part. Nevertheless their main focus is basically on the HPC and scientific workflows with none or minimum support of dynamic data analytics. On the other side

[114] Ewa Deelman, Tom Peterka, Ilkay Altintas, Christopher D. Carothers, Kerstin Kleesevan Dam, Kenneth Moreland, Manish Parashar, Lavanya Ramakrishnan, Michela Taufer, Jef frey S. Vetter: The future of scientific workflows. IJHPCA 32(1): 159-175 (2018)

[115] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole A. Goble, Matthew R. Pocock, Peter Li, Tom Oinn: Taverna: a tool for building and running workflows of services. Nucleic Acids Research 34(Web-Server-Issue): 729-732 (2006)

[116] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, R. Kent Wenger: Pegasus, a workflow management system for science automation. Future Generation Comp. Syst. 46: 17-35 (2015)

[117] Casey Robinson, Douglas Thain: Automated packaging of bioinformatics workflows for portability and durability using makeflow. WORKS@SC 2013: 98-105

systems such as Airflow[118], Luigi[119] and Argo[120] have been designed for Cloud applications and allow the usage of more flexible abstractions such as containerization and microservices which makes them more flexible and more adapted for data analytics. Nevertheless those systems can principally address batch processing operations with no facilitations for streaming operations.

Dealing with Streaming data is a need of new applications and use cases requiring reactivity and flexibility. Systems using IoT and Edge Computing will need to support stream processing to better cope with the challenges of new technologies and applications. Software solutions such as Beam[121] and Google Cloud Dataflow[122] have been designed to provide a unified model for batch and stream data processing with facilitation in the expression of Big Data workflows using higher level abstraction. They integrate seamlessly with specialized data processing frameworks such as Spark[123], Flink[124] and others on which they delegate the difficult task of runtime.

Those systems are ideal for Cloud Computing infrastructures with powerful computing characteristics but their design choices such as the choice of programming language does not make them adaptable for use cases implicating Edge Computing and IoT. Indeed, their choice of Java and Scala as base language makes them heavyweight for constrained computing power infrastructures.

3.1.2.3.1 Beyond state of the art

In the context of EMPYREAN, we will enhance RYAX's open-source workflow management system to support execution on multiple sites (Edge or Cloud) mainly supporting Kubernetes clusters, or different distributions of it (such as K3s). Furthermore RYAX will be integrated to support the open-source Zenoh-flow dataflow programming enabling users to define the internal dataflow of applications while supporting in a fine-grained manner real-time streaming executions. Finally the workflow-based application design will support the EMPYREAN concepts of Associations and Aggregators.

3.1.2.4 Cyber Threat Intelligence

Recent years have witnessed a dramatic growth in the amount of Cyber Threat Intelligence (CTI) available to the general public and to companies. This surge is driven by the increasing frequency and sophistication of cyber threats, necessitating robust and dynamic responses

118 "Apache Airflow: Programmatically Author, Schedule, and Monitor Workflows." Available: https://airflow.apache.org

119 "Luigi: A Python Module for Building Complex Pipelines of Batch Jobs." Available: https://github.com/spotify/luigi

120 "Argo: Workflow Engine for Kubernetes." Available: https://github.com/argoproj/argo

121 "Apache Beam: An Advanced Unified Programming Model." Available: https://beam.apache.org/

122 "Google Cloud Dataflow: Fully Managed Stream and Batch Data Processing." Available: https://cloud.google.com/dataflow

123 "Apache Spark: Unified Analytics Engine for Big Data Processing." Available: https://spark.apache.org/

124 "Apache Flink: Stateful Computations Over Data Streams." Available: https://flink.apache.org/

from the cybersecurity community. Platforms such as AlienVault OTX[125] and VirusTotal[126] play pivotal roles in this ecosystem by providing an ever-growing number of Indicators of Compromise (IoC) to the cybersecurity community.

Platforms for CTI: AlienVault OTX (Open Threat Exchange) and VirusTotal are among the most prominent platforms in the CTI landscape. AlienVault OTX enables users to collaborate and share threat data, enhancing collective cybersecurity awareness and defenses. Similarly, VirusTotal aggregates and analyzes files and URLs to identify viruses, worms, trojans, and other kinds of malicious content. These platforms are invaluable resources for security professionals who need to keep abreast of the latest threats and vulnerabilities.

Challenges in CTI Analysis: Despite the abundance of data provided by these platforms, the sheer volume and diversity of information make comprehensive analysis a daunting task. The data includes various types of IoCs, such as malicious IP addresses, domain names, file hashes, and URLs, which need to be continuously monitored and analyzed[127]. This task is further complicated by the heterogeneity of the data, as different sources may use different naming conventions, formats, and levels of detail.

Research Solutions: To address these challenges, the research community has developed various tools and methodologies. One significant focus has been on the distribution of malware and the usage of botnets. For instance, the study of malware distribution patterns helps in understanding how malicious software spreads across networks and infects systems. Research in botnet analysis[128] has provided insights into their structure, behavior, and the mechanisms used for command and control, which are crucial for developing effective mitigation strategies.

Heterogeneous Information Handling: The diverse nature of CTI data requires sophisticated tools to normalize and classify the information. AVClass[129] is one such tool that aims to assist security experts by providing automated malware labeling. By standardizing the labels used for different malware samples, AVClass helps in reducing the confusion and inconsistency that can arise from heterogeneous data sources. This standardization is critical for accurate threat analysis and response.

Knowledge Extraction from Unstructured Sources: Another significant area of research is the extraction of knowledge from unstructured data sources. Cybersecurity reports, threat intelligence blogs, and social media posts often contain valuable information but are not

[125] https://otx.alienvault.com/
[126] https://www.virustotal.com/
[127] Allegretta, M., Siracusano, G., Gonzalez, R., & Gramaglia, M. (2023). Are crowd-sourced CTI datasets ready for supporting anti-cybercrime intelligence?. *Computer Networks*, *234*, 109920.
[128] Massi, J., Panda, S., Rajappa, G., Selvaraj, S., & Revankar, S. (2010). Botnet detection and mitigation. *Student-Faculty Research Day, CSIS, Pace University, White Plains, NY (May 2010)*.
[129] Silvia Sebastián, Juan Caballero. AVClass2: Massive Malware Tag Extraction from AV Labels. In proceedings of the Annual Computer Security Applications Conference, December 2020.

structured in a way that can be easily analyzed by automated systems. Tools like Ladder[130] have been developed to address this issue by automatically labeling and organizing content from these unstructured sources. Ladder and similar tools leverage natural language processing (NLP) techniques to identify and categorize relevant information, making it more accessible for further analysis.

The ultimate goal of these efforts is to integrate structured and unstructured data to provide a comprehensive view of the cybersecurity landscape. By combining data from platforms like AlienVault OTX and VirusTotal with insights extracted from unstructured sources, security professionals can gain a more nuanced and complete understanding of emerging threats. This integrated approach allows for the creation of advanced analytical models and knowledge graphs that represent the current cybersecurity context in a more holistic manner.

3.1.2.4.1 Beyond state of the art

During the EMPYREAN project, we aim to significantly enhance the capabilities of cybersecurity professionals by collecting Cyber Threat Intelligence (CTI) from a variety of sources, including the Cyber Threat Alliance (CTA). The CTA, known for its high-quality threat intelligence sharing among industry leaders, will be a cornerstone of our data acquisition efforts. By integrating data from the CTA and other prominent sources, we will compile a comprehensive repository of Indicators of Compromise (IoCs), malicious IP addresses, domain names, file hashes, URLs, and more. This extensive dataset will provide a rich foundation for thorough threat analysis and proactive defence strategies.

A key component of the EMPYREAN project is the development of a user-friendly interface designed to streamline the information retrieval process for security experts. Recognizing the challenges posed by the sheer volume and diversity of CTI data, our interface will prioritize ease of use and efficiency. Security professionals will be able to quickly search, filter, and visualize relevant threat information, allowing them to stay ahead of emerging threats and vulnerabilities. The intuitive design will ensure that even those with varying levels of technical expertise can navigate the platform effectively, making critical threat intelligence accessible to a broader audience.

In addition to the user interface, the EMPYREAN project will focus on the development of advanced algorithms to identify trends and patterns within the CTI data. These algorithms will leverage machine learning and data mining techniques to analyze the vast amounts of collected data, uncovering insights that might otherwise go unnoticed. By identifying emerging threats and trends, our system will enable proactive measures and informed decision-making. This analytical capability will be crucial in understanding the evolving tactics, techniques, and procedures of cyber adversaries, thereby enhancing the overall cybersecurity posture of organizations.

[130] Alam, M. T., Bhusal, D., Park, Y., & Rastogi, N. (2023, October). Looking beyond IoCs: Automatically extracting attack patterns from external CTI. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses* (pp. 92-108).

3.1.2.5 Analytics-friendly IoT Erasure Coded Data Query

Erasure coding offers a highly cost-efficient way to improve availability and reliability in distributed storage, especially when compared to the industry-standard 3-way replication[131]. Maximum Distance Separable (MDS) codes in particular, provide redundancy at an optimal cost in terms of additional storage. Unfortunately, erasure coding introduces some encoding and decoding overhead, making data access slightly slower. Furthermore, in its conventional application, byte-level access to the original data is infeasible. Thus, to access even a small part of the original data, the entire erasure coded fragments must be retrieved and decoded. Given these limitations, erasure coding has traditionally been used mainly for long-term, large volume object/file/blob storage.

Internet of Things (IoT) devices produce large amounts of data. Sensors on devices such as robots, smart home appliances, cars or even agricultural machinery constantly monitor their surroundings and record this information, usually in a centralized system. Given the vast quantities of data, distributed cloud storage is a natural choice. Much of the data must be kept for long durations, making erasure coding a prime candidate for ensuring high availability and reliability at a low storage cost. Sensor readings in time series data are usually highly correlated, and thus a prime candidate for compression.

However, IoT data storage has an additional requirement that has traditionally been at odds with erasure coding and compression: it must be efficiently queryable. To extract value from IoT data, analytics workloads must be able to address individual parts (sometimes as small as a few bytes) with as little overhead as possible in terms of network transfers. This is especially true when using cloud storage, where data egress is prohibitively expensive.

3.1.2.5.1 Beyond state of the art

Thus, operators of state of the art IoT storage systems must make a choice between costly storage with efficient queries (replication) or cost-effective storage with highly inefficient queries (erasure coding). EMPYREAN will advance the state of the art by improving the efficiency of erasure coded queries through a novel set of data alignment, storage and retrieval techniques. We will devise, refine and implement a special storage schema, specifically designed for time series IoT data, with the goal of making byte-level access to erasure coded data traffic-efficient. This will greatly benefit scenarios where large volumes of data must be stored over a long period of time, combined with the need to run analytics workloads that only access a small portion of the entire volume.

Beyond erasure coding, compression is also widely used to reduce storage costs. It can be particularly effective when storing highly-correlated data, which is often the case with measurements coming from IoT sensors. Unfortunately, most state of the art compression schemes make queries unfeasible, in much the same way as erasure coding, by restricting byte-level access. EMPYREAN sets out to solve this problem and perform queries directly on

[131] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System," *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Village, NV, USA, 2010, pp. 1-10, doi: 10.1109/MSST.2010.5496972.

compressed time-series data. We plan to investigate the applicability of a novel family of compression schemes called Generalized Deduplication[132], some of which are specifically tailored to IoT data.

If done efficiently, running queries directly on erasure coded and compressed data would be a significant scientific achievement with tangible real-world benefits.

3.1.2.6 Application Deployment Based on Unikernels

Containers have become the cornerstone of modern application deployment, significantly influencing the cloud-native landscape. They encapsulate applications along with their dependencies, ensuring consistent environments across various stages of development and deployment. The popularity of containers is primarily due to their lightweight nature, efficient resource utilization, and rapid deployment capabilities. Key technologies like Docker and Kubernetes (K8s) have cemented containers as the go-to solution for scalable, flexible, and portable application management.

However, the security isolation provided by containers has its limitations. Containers share the same host operating system kernel, which makes them vulnerable to kernel-level attacks. Numerous Common Vulnerabilities and Exposures (CVEs) related to privilege escalation, where a malicious user can break out of the isolation boundaries of a container, appear frequently in security advisory bulletins[133]. This security concern necessitates additional measures to enhance isolation and security within containerized environments.

To mitigate the inherent security risks of containerization, container sandboxing mechanisms have been developed. Sandboxing enhances the isolation of containers by providing an additional layer of security, preventing malicious code within a container from affecting the host system or other containers. This is particularly crucial in multi-tenant environments where untrusted workloads run side by side[129].

In multi-tenant environments, where multiple users or services share the same infrastructure, the risk of security breaches increases. This lack of strong isolation necessitates the use of additional sandboxing mechanisms to ensure that each container operates in a securely isolated environment, thereby protecting against potential attacks[134].

Technologies like gVisor and Kata Containers provide such sandboxing mechanisms. gVisor, for instance, implements a user-space kernel that intercepts and processes system calls made by the container, thereby isolating it from the host operating system. Kata Containers, on the other hand, encapsulate containers within lightweight virtual machines, leveraging the strong isolation properties of VMs while maintaining the efficiency of containers[129]. These solutions

[132] Prasad Talasila and Daniel E. Lucani. 2019. Generalized Deduplication: Lossless Compression by Clustering Similar Data. In 2019 IEEE 8th International Conference on Cloud Networking (CloudNet). 1–4. https://doi.org/10.1109/CloudNet47604.2019.9064140

[133] Mainas, C., Plakas, I., Ntoutsos, G., & Nanos, A. (2024). Sandboxing Functions for Efficient and Secure Multi-tenant Serverless Deployments. In SESAME '24 Proceedings. ACM.

[134] Nanos, A., Plakas, I., Ntoutsos, G., & Mainas, C. (2024). Enabling Cloud-native IoT Device Management. In MECC'24 Proceedings. ACM.

aim to strike a balance between performance and security, offering enhanced protection without significantly sacrificing the agility and resource efficiency that containers offer.

Despite these improvements, the additional layers for isolation in sandboxing mechanisms lead to higher resource overheads compared to traditional containers[129]. Encapsulating an entire operating system within each microVM contributes to increased memory and storage consumption, and the startup times of microVMs can no longer compare to the rapid instantiation of containers.

Unikernels represent a significant advancement in the domain of lightweight, secure, and performant application deployment. Unlike traditional operating systems, unikernels compile applications with only the necessary OS components into a single binary, resulting in a minimalistic and highly optimized runtime environment. This approach reduces the attack surface, enhances security, and improves performance due to the absence of unnecessary components.

Recent projects like Rumprun[135], Unikraft[136], OSv[137], and IncludeOS[138] have made significant strides in improving the usability and maturity of unikernels. These projects offer frameworks and tools that simplify the building and deployment of unikernels, making them more accessible for mainstream adoption. Unikernels' ability to start almost instantaneously and their minimal resource consumption make them ideal for cloud-native and edge computing scenarios[130].

3.1.2.6.1 Beyond the state of art

EMPYREAN envisions extending the unikernel concept with the development of Cloudkernels, a comprehensive set of systems software components designed for the deployment of applications as unikernels in cloud-native environments. Cloudkernels aim to overcome two primary challenges associated with unikernels: the complexity of the build and deployment process, and the overhead imposed by traditional virtualization mechanisms.

By integrating advanced orchestration frameworks and tools, Cloudkernels will streamline the process of converting applications into unikernels, managing their lifecycle, and orchestrating their deployment across diverse cloud environments. This approach promises to significantly reduce the resource footprint and enhance the performance and security of deployed applications, especially in resource-constrained edge environments.

The move towards unikernels and Cloudkernels in application deployment offers a transformative shift in how applications are developed, deployed, and managed. The primary benefits include enhanced security through reduced attack surfaces, improved performance

[135] "Rumprun: A lightweight unikernel for running applications." Available: https://github.com/rumpkernel/rumprun

[136] "Unikraft: Lightweight Unikernels for Fast and Secure Application Deployment." Available: https://unikraft.org

[137] "OSv: The Operating System Designed for the Cloud." Available: https://osv.io

[138] "IncludeOS: Minimal, Resource-efficient Unikernel for Cloud and IoT Applications." Available: https://www.includeos.org

due to minimalistic runtime environments, and efficient resource utilization suitable for both cloud and edge computing.

Future research and development efforts should focus on further simplifying the build and deployment processes of unikernels, enhancing their integration with existing cloud-native tools, and expanding their applicability across a broader range of use cases. Additionally, addressing the interoperability and standardization challenges will be crucial to fostering wider adoption and ensuring seamless operation across diverse environments.

3.2 EMPYREAN Enablers

The EMPYREAN Enablers section highlights the key components and technologies crucial for building the EMPYREAN platform. These enablers provide secure identity management, real-time monitoring, efficient data transport, and intelligent workload management. By incorporating advanced technologies like decentralized identifiers, attribute-based encryption, blockchain, and trusted execution environments, these enablers ensure a secure, scalable, and efficient IoT-edge-cloud ecosystem. They address platform challenges, enabling seamless orchestration, robust security, and optimal resource utilization across various applications and use cases.

1. Secure Identity and Access Management and Attribute-Based Credential Management (EN_1)

- **Description**: Ensures secure and private identity management, data verification, and robust cryptographic foundation for managing privacy-preserving attribute-based credentials across the platform.
- **Components**:
    - Privacy and Security Manager (UMU): Manages cybersecurity, privacy-preserving techniques, and identity management using decentralized identifiers (DIDs) and verifiable credentials (VCs).
    - P-ABC (UMU): Provides a distributed privacy-preserving attribute-based credential system based on PS multisignatures.

2. Real-Time Monitoring, Observability, and Service Assurance (EN_2)

- **Description**: Enables real-time visibility into system performance and security, allowing for prompt response to anomalies and efficient resource utilization. Ensures that applications perform as intended by dynamically adjusting deployments based on real-time analytics and telemetry data.
- **Components**:
    - Telemetry Service (UMU): Handles observability and telemetry, allowing system administrators to understand system behavior and monitor performance metrics in real-time.
    - Telemetry Engine (ICCS, UMU, NEC, RYAX, NUBIS, CC): Maintains a global view of infrastructure resources and deployed applications, coordinating monitoring probes and providing historical telemetry data.

- Analytics Engine (ICCS, RYAX, UMU, ZSCALE): Implements service assurance mechanisms to detect infrastructure issues and deployed application performance, triggering dynamic adjustments.

3. High-Performance Data Transport Service (EN_3)

- **Description**: Facilitates reliable and efficient data transport using RDMA-based technologies, optimizing network I/O and computation overlap.
- **Components**:
  - Software-defined RDMA-based Unified Transport Service (NVIDIA, NUBIS): Provides a reliable RDMA-based transport service with ring-buffer lock-free structures and integration with large computational pipelines.

4. Decentralized Data Communication and Storage (EN_4)

- **Description**: Provides decentralized and distributed communication mechanisms with efficient pub/sub and data querying capabilities.
- **Components**:
  - Decentralised and Distributed Communication Layer (ZSCALE): Offers data dispatcher functionalities with efficient pub/sub and distributed queries, integrated into the Eclipse Zenoh open-source project.

5. Hybrid Cloud-Edge Storage and Efficient Time-Series Data Storage Management (EN_5)

- **Description**: Manages storage resources across cloud and edge environments, supporting hybrid policies for data distribution, redundancy, and security. Enhances the storage and retrieval of IoT time-series data using erasure coding for reliable data management.
- **Components**:
  - Edge Storage Gateway (Chocolate Cloud): Provides an access point to the EMPYREAN storage service through an industry-standard S3 interface, supporting hybrid storage policies.
  - Edge Storage (Chocolate Cloud): Provides a common abstraction of storage resources located at the edge, using a containerized version of MinIO.
  - IoT Query Engine (Chocolate Cloud): Stores and accesses IoT time series data using erasure coding for efficient data management.

6. Workflow Design and Management (EN_6)

- **Description**: Enables the design, deployment, and monitoring of data analytics workflows across heterogeneous computing environments.
- **Components**:
  - Ryax Workflow Engine (RYAX): Open-source workflow engine for designing, deploying, and monitoring data analytics workflows on cloud, edge, and HPC infrastructures.

7. Intelligent Workload Autoscaling (EN_7)

- **Description**: Utilizes AI/ML techniques to optimize workload autoscaling, ensuring efficient resource allocation and utilization.
- **Components**:

- AI-enabled Workload Autoscaling (RYAX): Enhances Kubernetes orchestrator with AI/ML techniques for intelligent resource requests and dynamic adaptation based on historical data.

8. Cyber Threat Intelligence (EN_8)

- **Description**: Provides insights into past cyber threat events to quantify and mitigate risks, enhancing overall platform security.
- **Components**:
  - CTI Analysis Module (NEC): Provides information about past Cyber Threat Intelligence events observed around the world, serving as an external source to quantify system risks.

9. Service Orchestration and Resource Management (EN_9)

- **Description**: Manages the orchestration and resource allocation for efficient service deployment and operation across the EMPYREAN infrastructure.
- **Components**:
  - EMPYREAN Orchestrator and Controller (ICCS, RYAX, NUBIS): Ensures efficient service orchestration and resource management, initiating application deployment and coordinating necessary actions.
  - Decision Engine (ICCS, RYAX): Implements multi-objective optimization and orchestration algorithms, interacting with the Service Orchestrator and Telemetry Service.

10. Resource Registration and Management (EN_10)

- **Description**: Manages the registration and tracking of IoT devices, edge, and cloud resources, facilitating seamless application deployment.
- **Components**:
  - EMPYREAN Registry (ICCS, RYAX, NUBIS, UMU): Manages the registration of IoT devices, edge, and cloud resources in Associations, tracking available services and resources.

11. Association Management and Coordination (EN_11)

- **Description**: Manages and coordinates the operation of EMPYREAN Associations, ensuring efficient workload deployment and resource management.
- **Components**:
  - EMPYREAN Aggregator (ICCS, UMU, CC, ZSCALE, RYAX, NUBIS): Manages and coordinates the operation of an EMPYREAN Association, including core services for orchestration and resource management.

12. Hardware Acceleration and Security (EN_12)

- **Description**: Enables the use of hardware accelerators for compute-intensive tasks while ensuring data security and integrity.
- **Components**:

- vAccel (NUBIS): An open-source framework for mapping hardware-accelerate-able workloads to relevant hardware functions, enhancing security and performance.

13. Unikernel Application Development (EN_13)

- **Description**: Simplifies the development and deployment of unikernel-based applications, reducing engineering overhead and ensuring efficient execution.
- **Components**:
  - Application Builder for Unikernels (NUBIS): Addresses the deployment of applications in cloud-native environments using unikernels, simplifying the building and deployment process.

14. Multi-Environment and Multi-Architecture Application Packaging (EN_14)

- **Description**: Streamlines the packaging process for applications, creating OCI-compatible container images and supporting multiple architectures and programming languages for deployment across various execution environments.
- **Components**:
  - Application Packaging (NUBIS): Tool for bundling binary artifacts into OCI container images for deployment across EMPYREAN's supported execution modes.
  - NIX Based Environment Packaging (RYAX): Tool for performing multi-arch and polyglot environment packaging to build components for workflows.

15. Versatile Container Runtime Integration (EN_15)

- **Description**: Facilitates the deployment of applications in various execution environments, integrating unikernels with container runtimes.
- **Components**:
  - Container Runtime (NUBIS): Runtime capable of spawning unikernels and integrating them with generic container runtimes compatible with Kubernetes and serverless architectures.

16. Secure and Trusted Execution (EN_16)

- **Description**: Establishes a secure execution environment with measured boot mechanisms, supporting scalable and trusted operations across the IoT-edge-cloud continuum.
- **Components**:
  - Secure Execution Environment (NUBIS): Focuses on secure and trusted execution across the IoT-edge-cloud continuum, supporting secure and measured boot mechanisms.

17. Optimized Task Scheduling (EN_17)

- **Description**: Implements scheduling algorithms to minimize cold start delays and optimize the placement of tasks based on container layer locality.
- **Components**:
  - Layers Locality Scheduler (RYAX): K8S scheduling algorithm to minimize cold start delays and optimize task placement based on container layers.

3.3 EMPYREAN relation with ongoing relevant EU projects

Next, we describe the relationship between EMPYREAN and ongoing EU projects in similar topics.

3.3.1 HORIZON EUROPE MLSYSOPS (2023-2025)

Common Partners: CC, NVIDIA, NUBIS

MLSYSOPS will build an AI-based framework for autonomous E2E system management across the edge-cloud continuum. Some of the orchestration and adaptation features developed in MLSYSOPS could be reused and extended in EMPYREAN. For example, CC is developing a mechanism to migrate data between different storage policies.

3.3.2 HORIZON EUROPE RESCALE (2024-2026)

Common Partners: CC

The overarching goal of RESCALE is to systematically analyze and extend, as necessary, every hardware and software layer in a computing system and apply novel tools and methodologies at every step of the entire supply chain. Any security improvements to CC's code base will be carried over to EMPYREAN. Some of the DevOps tools created as part of RESCALE will also be employed to aid in implementing a novel storage solution for EMPYREAN.

3.3.3 IPCEI EUROPE E2CC (2024-2027)

Common Partners: RYAX

The E2CC project intends to deliver an end-to-end standardized integration layer enabling interoperability and creating a continuum from the Edge to the Cloud. It will provide the core technological referential for interconnecting with Cloud providers, including Cybersecurity, Decarbonization, Orchestration and Platform functions. Among others the project will deal with Edge to Cloud orchestration (to ensure energy efficiency, security, interoperability) along with middleware & system SW optimization related to Federation and multi-cloud meta-orchestration. Furthermore, the project will work on Advanced Smart Data Processing through end-to-end security solutions for edge to cloud and the development of MLOps and AI services adapted to the Edge to Cloud. EMPYREAN will relate to this IPCEI project as it has been specifically mentioned in the DoA and RYAX who participates in both projects will enable the exchanges and knowledge transfer between the projects especially in the areas around orchestration upon which RYAX will be contributing in.

3.3.4 HORIZON EUROPE FLUIDOS (2024-2026)

Common Partners: UMU

FLUIDOS aims to create a flexible, scalable, and secure decentralized operating system for the edge-cloud continuum. This project will integrate advanced edge computing capabilities to optimize resource utilization and improve data processing efficiency across distributed networks. One of the key innovations of FLUIDOS is the REAR protocol (REsource Advertisement and Reservation), designed for dynamic resource discovery and reservation in IoT environments.

The REAR protocol will enable EMPYREAN to dynamically acquire resources between associations of IoT devices, enhancing its ability to manage and utilize distributed computing resources effectively. The protocol supports automatic and autonomous resource discovery and integration, ensuring seamless operations across various domains.

4 Use Cases Analysis

4.1 Anomaly Detection in Robotic Machining Cells (UC1)

4.1.1 Overview

The objective of this use case is to develop a system able to perform real-time process monitoring in robotic machining cells performing composite manufacturing operations using high-frequency data. These operations include turning, milling, and drilling. Process monitoring involves real-time signal monitoring and alerting of abnormal machining operations.

The goal is to create an application that enables quick identification and response to any deviations in the machining process by timely processing real-time data from high-frequency sensors to improve production efficiency and minimize losses.

4.1.2 Detailed Description

In the manufacturing landscape, the adoption of robots for machining tasks significantly boosts flexibility and reduces costs compared to traditional machine tools. The usage of robots allows for rapid adjustments to production processes and product designs. This adaptability is especially beneficial in dynamic markets, where manufacturers need to quickly respond to changing consumer demands and technological advancements, while maintaining high precision and efficiency. However, introducing robots in machining requires rigorous process monitoring to manage issues like precision loss or tool breakage among common process-related defects when machining composite materials. Addressing these challenges is essential for improving efficiency and maintaining quality standards in the manufacturing industry.

The use case "Process Monitoring and Anomaly Detection in Robotic Machining Cells" focuses on developing a system capable of real-time monitoring of machining operations in robotic cells machining composite materials. Using high-precision sensors, critical variables are measured to create process fingerprints and detect anomalies in real-time.

For signal monitoring and analysis, IDEKO has a robot equipped with a Smart Box, an industry-ready IoT edge device developed by IDEKO to collect data from machines/robots. Additionally, the robot used for this use case will include high-frequency data acquisition equipment such as the INGETEAM IC2/IC3 models, for a more thorough analysis of the machining processes performed by the robots.

The use case proposes a scenario with the following key points:

- Simulate a scenario with a client with three machining cells being monitored at the same time.
- Use data for the existing robot at IDEKO facilities.
- Simulate data for additional robots based on real data from the existing robot.

- Each robot will be attached to a Smart Box IoT Data Gateway.
- Client may have edge devices (will be adapted on-demand for the project).
- Machining data will be simulated using real data generated at some IDEKO's robotic machining cell.
- An application will be deployed into the available infrastructure for real-time process monitoring.
- The scenario will simulate different workloads, hence demands variable computing power requirements.
- The application will use AI techniques for anomaly detection:
    - Techniques are currently being tested at IDEKO and several approaches are currently being analyzed)
    - Techniques may not be only based on ML algorithms
- The available platform must respond to application variable workloads to achieve application KPIs and requirements.
- UC will allow the application to move from current offline analysis to online analysis.
- UC will allow to process data in high frequency vs the current approach that uses low frequency data.

The use case full scenario is depicted in the image below.

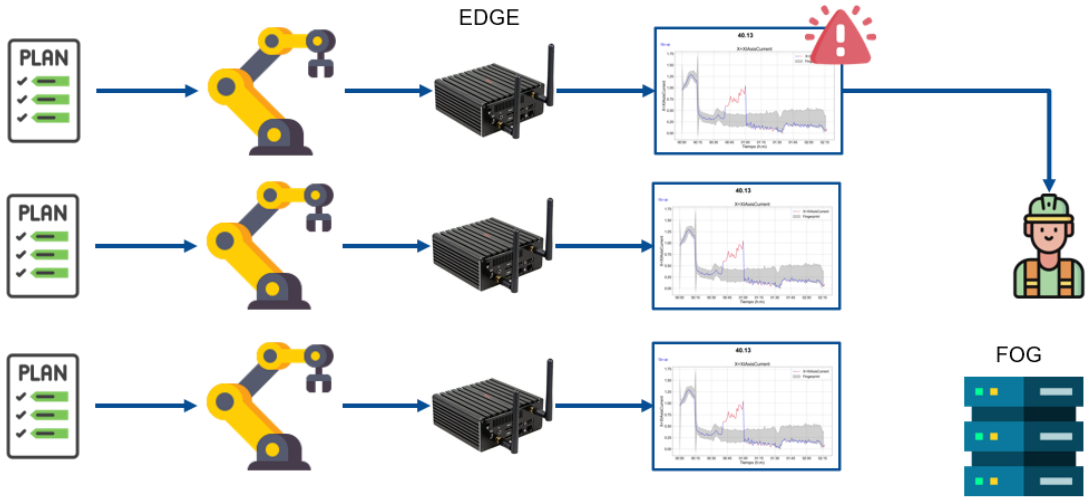

Figure 6: This use case proposes a scenario of a client with 3 robots

Collaboration with EMPYREAN will leverage platform-based associations, providing a secure environment for interaction between robots and peripheral resources, as well as orchestration and automatic scaling mechanisms to manage processing and storage demands.

Data to be used

IDEKO has a robotic machining cell operated by a single robot. Machining tests will be carried out in the cell and data retrieved from those operations will be used to simulate the described scenario. Figure 7 illustrates the different external sensors attached to this robot.

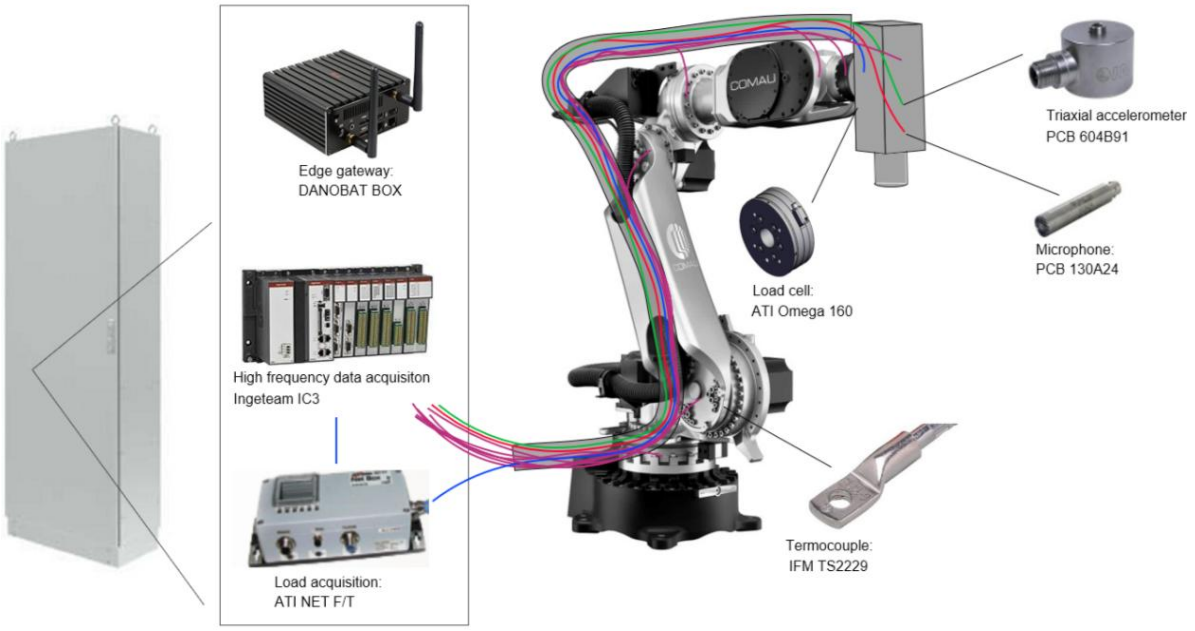

Figure 7: Different external sensors attached to a robot

Each robot will have:

- A force measuring sensor
- A triaxial vibration sensor
- A microphone
- Temperature sensors on every joint

Based on this setup, the following table describes the tentative setup data to be used by the application for process monitoring.

Table 1: Data sources and data acquisition frequency

DATA SOURCE	COMMUNICATION PROTOCOL	DATA ACQUISITION FREQUENCY
CNC Internal variables	S7 (SIEMENS CNC)	1Hz
Vibration	ModBus TCP	51200 Hz
Force	ModBus TCP	1000 Hz
Sound	ModBus TCP	51200 Hz
Temperature	ModBus TCP	1000 Hz

4.1.3 Current State - Future State with EMPYREAN

In the current state, manufacturing orders are initially planned and divided into operations to be executed by the robot. Once the robot completes the machining part, data from selected operations undergo offline analysis through the workflow application developed by IDEKO for anomaly detection in machining processes. If the analysis detects an anomaly the operator is notified (Figure 8).

Figure 8: The high-level description of the offline analysis

The workflow application for offline process monitoring relies on low-frequency data. These applications enable the analysis of individual operations by creating a unique fingerprint for each operation using statistical methods based on low-frequency data. The system notifies the operator after the manufacturing operation has been already finished (offline operation). In these applications, manufacturing processes are segmented into individual operations in order to analyze similar operations collectively. To efficiently handle the data, data processing pipelines are implemented using a microservices architecture, allowing for modular and scalable processing of the collected data.

Figure 9 presents the workflow of the current application implementation for anomaly detection, including also the queues used for intercommunication between processes.

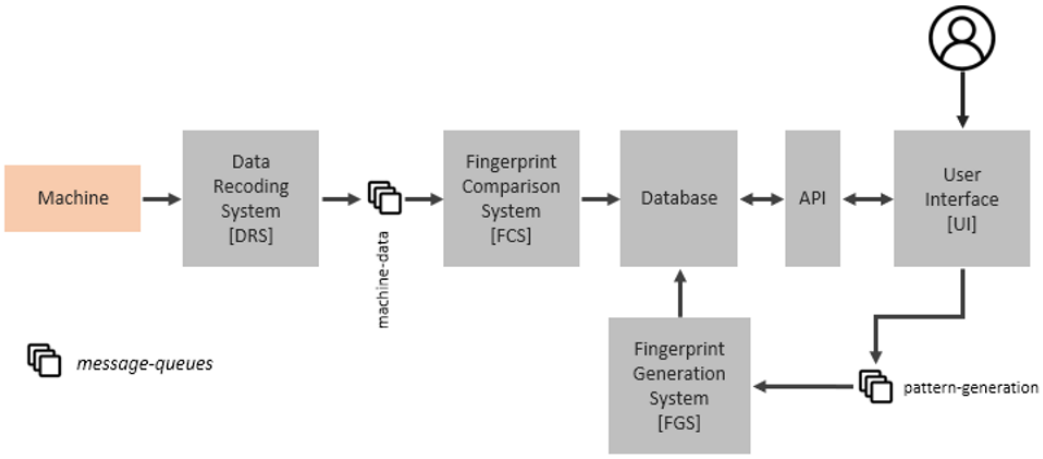

Figure 9: The workflow application for offline process monitoring

The role of each component is described below:

- The *DRS component* is responsible for monitoring the execution of operations and recording the associated data. This data is sent to the *machine-data* queue.
- The *FCS component* listens to the *machine-data* queue and, if there is a pattern for that piece model, it executes the comparison of the piece data against the pattern and stores the results in the database.
- The user interface consumes the *API* to present everything necessary in the interface.
- From the *User interface*, the user can generate patterns. In that case, a generation task is queued in the *pattern-generation queue*.
- The *FGS consumes the pattern-generation queue* and for each message found, it generates a pattern with the data of the selected pieces.

The current state provides a basis for monitoring and analyzing manufacturing processes, but it could be improved in several ways:

- Integrating real-time monitoring (online process) capabilities.
- Leveraging high-frequency data for more precise analysis and decision-making.
- Sharing the available infrastructure among the assets being monitored.

This use case concentrates on tackling the previously mentioned concerns via the EMPYREAN Associations-based continuum. An Association will provide a secure and trusted execution environment for the interactions between robots and edge resources. Additionally, resource orchestration and autoscaling mechanisms will efficiently and dynamically handle the increased demands within or across multiple Associations for processing and storage in a multi-agent manner.

Our future state objective is threefold:

- Firstly**,** to transition to an online monitoring system;
- Secondly, to detect anomalies before the completion of each operation;
- And thirdly, to leverage mid to high-frequency data for analysis.

The application workflow will remain the same, but the way the services are managed, offloaded and orchestrated will be delegated to the EMPYREAN framework. The IDEKO use case will utilize mechanisms and tools developed in WP3 and WP4, which will be exposed through the EMPYREAN SDK to address the identified challenges, as described in the following section.

4.1.4 Challenges to be Addressed

The implementation of real-time monitoring systems faces various technical challenges, particularly the need to process large amounts of data within relatively short timeframes, given the limited computing power of with edge devices. The edge-cloud continuum, which integrates cloud computing with multi-layered edge processing resources, emerges as a promising solution by offloading, orchestrating, and balancing computing tasks among the available resources. Consequently, the main challenges to be addressed include:

- **Lack of real-time capability** to identify optimal variables amidst the diverse operations, such as drilling and milling, throughout the machining process.
- **Potential insufficient computing power** to handle the extensive list of operations, hinders the generation of normality patterns and timely comparisons.
- **Variety of data sources** (e.g., robots with sensors) with high frequency need to be efficiently transferred, stored and processed timely.

4.1.5 Deployment Environment

The proposed architecture to support the use case is based on the typical architecture found in a machine tool client. It is composed of two layers: the on-device edge resources associated with the robots and the on-premise edge resources set up by the client to host more demanding workloads.

This use case proposes a scenario where a client with 3 robots that aims to deploy the process monitoring solution. Each robot is equipped with edge devices, providing varying CPU, RAM, and storage (HD) models, offering diverse range of computational capabilities, as illustrated in the table below. It is worth noting that none of these devices are GPU-powered. They support container-based application deployment, which facilitates efficient and flexible deployment of software components. Additionally, the client has additional and more powerful edge resources in its facilities.

Table 2: Characteristics of the deployment environment resources

DEVICE	CPU	RAM	HD
Edge Smart Box	Intel Celeron 1.83GHz	4GB	120GB
Edge Smart Box	Intel Celeron 1.83GHz	8GB	480GB
Edge Smart Box	Intel Celeron 1.83GHz	2GB	32GB
On premises edge devices	*Number of resources and characteristics to be defined during the project*		

4.1.6 KPIs

The success of the anomaly detection use case in robotic machining cells will be assessed using several Key Performance Indicators (KPIs). These KPIs are designed to evaluate the effectiveness of the deployed system in meeting its objectives and to provide actionable insights for continuous improvement. They focus on enhancing operational efficiency and responsiveness within our manufacturing processes. The KPIs for this use case are as follows:

No	Indicator	Success Criteria
1	Transition from offline operation analysis to real-time operation fingerprint analysis	-
2	Ability to process real-time data	3 robots / 200 operations
3	Ability to alert an abnormal operation	máx. 2 sec after it occurs

To accurately assess these KPIs, some initial evaluation metrics have been established.

1. **Transition from offline operation analysis to real-time operation fingerprint analysis:** Currently, anomaly detection is performed offline, after operations have been completed. The goal is to transition to a system capable of analysing operation fingerprints in real time, while operations are still running. This will enable earlier detection of deviations and support more responsive decision-making.

2. **Ability to process real-time data:**

This KPI will be assessed based on the system's ability to handle continuous data streams from multiple robotic machining cells. The initial target is to support 3 robots and 200 operations, ensuring low latency and high data integrity throughout the data workflows.

3. **Ability to alert an abnormal operation:**

The system must be capable of issuing an alert within a maximum of 2 seconds after an anomaly is detected. This requires fast detection algorithms, optimized data handling, and reliable notification mechanisms that enable timely intervention before part quality is affected.

4.1.7 Validation and Testing

The proposed deployment environment includes evaluation scenarios where edge devices lack the necessary computing power to meet computational demands, thus necessitating the orchestrated execution and offloading of workflows across the available resources. Anomalies will be forced based on real machining data, such as unexpected changes in vibration patterns or abnormal temperature fluctuations.

During the validation and testing phase, we will use the three key performance indicators. Firstly, we will evaluate the transition from offline operation analysis to real-time operation fingerprint analysis. Then, we will measure the system's ability to process real-time data from three robots and a defined number of operations. Lastly, we will verify the system's ability to alert about an abnormal operation within a maximum of 2 seconds after it occurs.

4.2 Proximal Sensing in Agriculture Fields (UC2)

4.2.1 Overview

This use case focuses on the dynamic assessment of Soil Organic Carbon (SOC) to evaluate soil conditions in agricultural fields. By leveraging proximal sensing technology coupled with edge computing, this system enables real-time analysis and efficient SOC assessment without the need for centralized data processing. This approach fosters integrative farm management and supports sustainable agricultural practices.

4.2.2 Detailed Description

Traditional methods for assessing soil properties, such as laboratory analysis face numerous challenges in terms of efficiency, timeliness, and accuracy. These methods often involve time-consuming processes that can delay critical decisions and actions in farm management. Additionally, the results from different laboratories may not be harmonized, leading to inconsistencies and difficulties in comparing data from various sources.

The EMPYREAN Associations-enhanced continuum addresses these challenges by integrating proximal sensing technology with edge computing to enable dynamic and efficient SOC assessment. Proximal sensing involves using sensors placed close to the soil to measure various properties, such as moisture, temperature, and organic carbon content. These sensors can be mounted on various platforms, including Unmanned Aerial Vehicles (UAVs), robots, and tractors, allowing for flexible and comprehensive soil monitoring across large agricultural fields. Edge computing plays a critical role in this system by processing data locally at the edge of the network, near the data source, rather than relying on a centralized server. This reduces latency and enables real-time data analysis, which is crucial for making timely decisions in farm management. By analyzing data from sensors and other information sources in near real-time, the system provides a continuous and up-to-date soil health assessment.

The EMPYREAN platform enhances this process by supporting distributed AI mechanisms, allowing machine learning (ML) models to be trained on data collected from various sensors directly in the field. Although initial training of these models requires soil samples and laboratory analysis to establish baseline data, once trained, the models can use real-time sensor data to provide accurate SOC predictions on-site. This approach ensures data privacy, reduces the need for large data transfers, and maintains high accuracy in soil assessments. Integrating these technologies through the EMPYREAN platform can revolutionize soil health monitoring and management. By enabling real-time, accurate, and efficient SOC assessment, the system supports sustainable agricultural practices, such as optimized fertilization and irrigation, which can lead to increased crop yields and reduced environmental impact. Monitoring soil health continuously and dynamically allows farmers to respond quickly to changes in soil conditions, ensuring that crops receive the right nutrients at the right time.

Furthermore, the platform's capability to handle large volumes of data and perform complex processing tasks at the edge addresses one of the significant challenges in proximal sensing. Traditional methods struggle with the sheer amount of data generated by high-resolution sensors, but the combined use of edge computing and distributed AI ensures that this data can be processed efficiently and effectively. This approach not only enhances the accuracy of soil assessments but also makes the system scalable and adaptable to different agricultural environments.

In summary, the EMPYREAN Associations-enhanced continuum leverages advanced technologies to overcome the limitations of traditional soil assessment methods. By combining proximal sensing with edge computing and distributed AI, the platform will provide a robust and efficient solution for dynamic SOC assessment, paving the way for more sustainable and productive agricultural practices.

4.2.3 Current State - Future State with EMPYREAN

Currently, the assessment of soil health and fertility conditions involves several methods. Traditional methods primarily include soil samples and laboratory analysis. Soil samples are collected from various locations within a field and sent to laboratories where they are analyzed for SOC and other properties. This process, while accurate, is labor-intensive, time-

consuming, and often costly. In addition to traditional laboratory analysis, other modern techniques are also used. Spectral proximal sensing involves using reflectance spectroscopy in the visible and near-infrared spectrum (Vis-NIR) to assess soil properties (Figure 10)[139]. This method provides rapid and non-destructive analysis but requires sophisticated equipment and expertise to interpret the data accurately. Remote sensing and the analysis of Earth Observation data represent another modern approach. [140]Satellite imagery and aerial photography are used to monitor large-scale agricultural fields. This method can cover extensive areas and provide valuable insights into crop health and soil conditions. However, the spatial resolution of satellite data can be limited, and cloud cover can obstruct visibility, making it less reliable for frequent monitoring. Despite the advancements these modern techniques offer, they still face challenges related to data harmonization and consistency. Results from different methods and sources may vary, making it difficult to create a unified and accurate soil health assessment. Additionally, these methods often rely on centralized data processing, which can introduce delays and reduce the timeliness of the information provided to farmers.

Figure 10: PSR+ field spectrometer

With the implementation of the EMPYREAN platform, the future state of soil health assessment will see a significant transformation in workflows. Integrating various sensors and data types, combined with edge computing, will enable dynamic and efficient SOC assessment, allowing for near real-time analysis of soil conditions. This will create a more responsive and adaptive agricultural management system. In the future state envisioned with EMPYREAN, UAVs equipped with advanced sensors will play a crucial role. Sensors mounted

[139] Angelopoulou, T., Balafoutis, A., Zalidis, G., & Bochtis, D. (2020). From laboratory to proximal sensing spectroscopy for soil organic carbon estimation—A review. *Sustainability*, *12*(2), 443.

[140] Castaldi, F., Hueni, A., Chabrillat, S., Ward, K., Buttafuoco, G., Bomans, B., ... & van Wesemael, B. (2019). Evaluating the capability of the Sentinel 2 data for soil organic carbon prediction in croplands. *ISPRS Journal of Photogrammetry and Remote Sensing*, *147*, 267-282.

on these UAV will estimate crop biomass[141], create management zones, and support crop protection activities, such as weed control. This aerial perspective allows for comprehensive coverage of agricultural fields, providing detailed data on crop health and soil conditions. Robotic sensors will further enhance soil assessment by performing detailed analyses in the identified management zones. The robot, equipped with visible and near-infrared (Vis-NIR) spectrometers, RTK GPS, and soil moisture sensors, will conduct dynamic and efficient SOC assessments directly in the field. The data collected will be processed on-site using edge computing, ensuring timely and accurate results. The integration of these advanced technologies will also support the creation and utilization of prescription maps. These maps will guide variable rate fertilization, ensuring that nutrients are applied precisely where needed, based on real-time soil health data. This targeted approach to fertilization will enhance crop yields, reduce waste, and minimize environmental impact.

The initial workflow conceptualized within EMPYREAN includes several key steps (Figure 11). First, UAVs with multispectral or hyperspectral cameras will be used to estimate cover crop biomass and identify management zones. These UAVs will also assist in crop protection management activities, such as weed control. Next, robots equipped with Vis-NIR spectrometers and other sensors will perform detailed soil assessments in the identified zones. The data collected will be processed using edge computing, enabling real-time SOC assessment. Once the soil assessments are completed, prescription maps will be generated using a Geographic Information System (GIS) platform. These maps will guide farmers in applying fertilizers precisely where needed, optimizing nutrient use, and enhancing crop health. The prescription maps will be transferred to fertilization equipment, ensuring accurate and efficient application. The EMPYREAN platform will support the orchestration of all these activities, managing the data flow and ensuring scalability and seamless integration between different components. By combining cloud computing for large-scale data processing and edge computing for real-time analysis, the platform will provide a robust and efficient solution for dynamic SOC assessment.

[141] Steenwerth, K., & Belina, K. M. (2008). Cover crops enhance soil organic matter, carbon dynamics and microbiological function in a vineyard agroecosystem. Applied soil ecology, 40(2), 359-369.

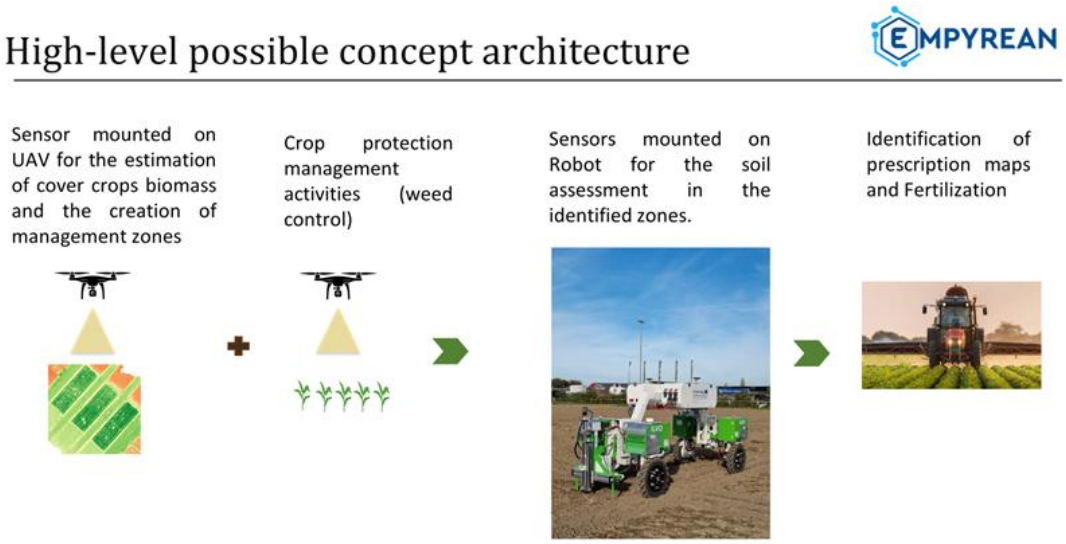

Figure 11: High-level description of possible concept architecture for the UC

It is important to note that these processes are envisioned as part of the initial project scope and may evolve as the project progresses. Continuous improvements and adjustments will be made based on ongoing research, technological advancements, and practical field experiences. This flexibility ensures that the EMPYREAN platform remains adaptable and effective in addressing the dynamic needs of modern agriculture.

In conclusion, the UC's future state with EMPYREAN represents a significant advancement in soil health monitoring and management. By leveraging proximal sensing technology, edge computing, and advanced AI mechanisms, the platform will enable real-time, accurate, and efficient SOC assessment. This will support sustainable agricultural practices, optimize resource use, and enhance crop yields, paving the way for a more productive and environmentally friendly agricultural system.

4.2.4 Challenges to be Addressed

Implementing proximal sensing for dynamic SOC assessment in agriculture fields introduces several challenges that must be addressed to ensure the system's effectiveness and efficiency.

Data Volume and Management: One of the primary challenges is the large volume of data generated by high-resolution sensors. Each sensor can continuously produce substantial amounts of data, leading to significant storage and processing requirements. Efficiently managing and analyzing this data in real-time is crucial for making timely and accurate decisions.

Networking Connectivity: Rural areas, where most agricultural fields are located, often suffer from poor networking connectivity. This makes it challenging to rely on cloud-based solutions that require constant and fast internet access. Edge computing helps mitigate this issue by processing data locally, but the initial setup and occasional synchronization with central servers still depend on network availability.

Integration of Multiple Data Sources: The system must integrate data from various sensors, including UAV-mounted cameras, soil moisture sensors, spectrometers, and more. Each type of sensor produces data in different formats and resolutions, necessitating sophisticated data fusion techniques to combine and interpret this information effectively.

Energy Consumption: Operating multiple sensors and processing units, especially in remote areas, requires a reliable power source. Monitoring and managing the energy consumption of these devices is essential to ensure continuous operation. Battery life, energy efficiency, and the ability to harvest energy from renewable sources are critical factors to consider.

Security and Data Privacy: With the use of distributed and edge computing, ensuring the security and privacy of the data becomes paramount. Measures must be in place to protect sensitive data from unauthorized access and cyber threats. Implementing robust encryption, secure communication protocols, and regular security updates are essential components of the system's design.

Scalability and Adaptability: The system must be scalable to handle varying field sizes and adaptable to different crops and soil types. Developing flexible and modular solutions that can be easily expanded or adjusted based on specific agricultural needs will ensure the system's long-term viability and effectiveness.

By addressing these challenges, the EMPYREAN platform aims to provide a robust and efficient solution for dynamic SOC assessment, ultimately leading to more sustainable and productive agricultural practices. Continuous research, development, and field testing will be necessary to refine the system and ensure it meets the diverse needs of modern agriculture.

4.2.5 Deployment Environment

The deployment environment for the proximal sensing use case in agriculture fields involves a sophisticated setup of devices, equipment, and technologies to enable dynamic and efficient SOC assessment. The exact deployment environment may evolve during the project's development, but the initial indicative setup includes a range of advanced tools and systems to support the use case. This use case will be validated within part of the EV ILVO field infrastructures, utilizing five indicative fields with various crops: Potatoes (4 ha), Winter rye (3 ha), Agroforestry (1 ha), Beets (3 ha), and Maize (5 ha). These diverse field conditions provide an ideal testing ground for deploying and validating the EMPYREAN platform's capabilities.

Possible Devices and Equipment: The deployment will utilize Raspberry Pi 4 or NVIDIA edge devices for processing data collected by drones and sprayers. For aerial data collection, two types of UAVs will be considered: the DJI Matrice 600 Pro and the DJI Matrice 350 RTK. These UAVs will be equipped with RGP, or multispectral and/or hyperspectral cameras, as well as external RTK GPS units, to capture high-resolution images and precise positioning data. Additionally, a visible and near-infrared (Vis-NIR) spectrometer will be used to measure soil reflectance properties. A EV ILVO robot equipped with a soil moisture measurement sensor and the spectrometer will be utilized to assess soil moisture content and reflectance

properties directly in the field. A tractor will be used for fertilization, guided by the prescription maps generated from the SOC assessments and other collected data.

Communications: The communication setup involves several technologies to ensure seamless data transfer and device coordination. WiFi will be used for communication between the UAV, robot, and the on-premises data center or the edge device. A 4G connection will be utilized for broader connectivity needs, especially in remote areas.

Power Sources: Power management is crucial for the deployment environment. The UAVs will be powered by drone batteries, which also supply power to the GPS and cameras during flight operations. Laptops used for data analysis and monitoring will rely on their internal batteries. The energy consumption of these devices will be closely monitored using battery monitoring systems that track voltage, current, and temperature, allowing for accurate estimation of energy usage. Possible power meters, such as Shelly Energy meters for laptops and shunt energy meters for external drone batteries, will measure energy consumption in watt-hours (Wh).

Energy Consumption: Comparing the energy consumption during the UC is essential to determine the most efficient deployment strategy. By monitoring battery levels and utilizing manufacturer specifications for each device, the deployment environment aims to optimize energy use and ensure sustained operations in the field.

Data Types: Various data types will be collected and processed in this deployment environment. Image analytics will involve images captured by UAVs equipped with RGB, multispectral and/or hyperspectral cameras, stored in JPEG, PNG, and RAW formats. Positioning data from RTK GPS units will be formatted as ASCII and follow the NMEA message structure standard. Soil reflectance data collected by spectrometers will also be in ASCII format. Additionally, environmental data, including moisture, temperature, and weather information from a weather station at ILVO, will be gathered in numerical data formats and stored in CSV files.

This deployment environment represents a comprehensive and technologically advanced approach to SOC assessment in agriculture fields. While the setup described is indicative and subject to change, it provides a robust framework for integrating proximal sensing, edge computing, and AI technologies to enhance soil health monitoring and management. Continuous refinement and adaptation of the deployment environment will be essential to address emerging challenges and leverage new technological advancements throughout the project's lifecycle.

4.2.6 KPIs

The success of the proximal sensing use case in agriculture fields will be measured through several Key Performance Indicators (KPIs). These KPIs will help evaluate the effectiveness of the deployed system in achieving its objectives and provide insights for continuous improvement. The primary KPIs for this use case are:

No	Indicator	Success Criteria
1	Development of processes that support the transition from subjective to objective, accurate and harmonised soil health data sets	2 SOC developed for different sensing technologies
2	Transition to a real- or near real-time assessment of soil, and water parameters, allowing cooperated integrated farm management;	1 SOC model should be able to run on edge hardware in near-real time
3	Reduce the time and effort needed to develop soil data-driven models, compared to training models with data from manual soil sampling campaigns.	by 25%

To accurately assess these KPIs, some initial evaluation metrics have been established.

1. **Development of Processes that Support the Transition from Subjective to Objective, Accurate, and Harmonized Soil Health Data Sets:**

To achieve this KPI, two Soil Organic Carbon (SOC) models will be developed using different sensing technologies, ensuring data harmonisation and comparability. The first SOC model will be based on UAV-acquired multispectral imagery, processed through cloud infrastructure. The UAV imagery will be stitched into orthomosaics, which will then serve as input for an AI model trained to estimate SOC levels across agricultural fields.

The second SOC model will operate using proximal sensing data captured by a spectrometer mounted on ILVO's autonomous field robot. This model will run on edge hardware, allowing immediate data analysis in the field. By integrating both approaches, the process moves from subjective, manually sampled measurements toward objective, sensor-based and harmonised datasets. This dual-model approach also enables cross-validation between aerial and proximal data sources, ensuring higher accuracy and robustness in SOC estimation.

2. **Transition to Near Real-Time Assessment of Soil Health**

This KPI will be achieved by deploying and optimising the edge AI model (based on spectrometer data) to operate in near-real time on the ILVO robot. During robot operation, the spectrometer continuously captures reflectance data from the soil surface. The on-board AI model processes this data locally, generating near-instant SOC assessments without requiring cloud connectivity.

The computational pipeline will be designed for efficiency, making use of lightweight model architectures, hardware acceleration, and optimised data transfer between sensors and processing units. The real-time insights produced by the robot can then be integrated into farm management systems to support adaptive decision-making, thus enabling coordinated and data-driven farm operations.

3. **Reduce the Time and Effort Needed to Develop Soil Data-Driven Models**

Traditionally, SOC models are trained using large datasets derived from manual soil sampling and laboratory analysis, which are time-consuming and expensive. This KPI aims to reduce this

effort by at least 25% through an iterative AI-based approach that leverages the synergy between the UAV and spectrometer-based models.

Specifically, the SOC assessment model running on the spectrometer will be used to (re-)train and refine the UAV-based model. Field data collected autonomously by the robot will provide continuous updates and ground-truth estimates that improve the UAV model's predictive performance. This creates a semi-automated feedback loop, where UAV imagery offers spatial coverage and the proximal spectrometer provides calibration and validation data. The result is a faster, less labour-intensive model development cycle that maintains high accuracy without the need for extensive manual sampling campaigns.

4.2.7 Validation and Testing

The current phase of the use case involves developing a high-level demonstration architecture and conceptualizing various key processes within the use case. These processes will leverage advanced technologies and EMPYREAN platform components to ensure seamless integration, efficient data processing, and effective soil health management. It is important to note that these are not final but represent an initial conceptualization that may evolve as the project progresses.

UAV and creation of management zones: In this process, a UAV equipped with an RGB multispectral or hyperspectral camera and RTK GPS will be used for several purposes. The primary tasks include estimating cover crop biomass, creating management zones, and managing crop protection activities such as weed control. The UAV will utilize pre-trained ML models to classify cover crop biomass based on the collected images. Additionally, the integration of Earth Observation (EO) data is under consideration, which could further enhance the accuracy and comprehensiveness of the assessments.

Robot, spectrometer and SOC assessment: This process involves a robot equipped with a visible and near-infrared (Vis-NIR) spectrometer, RTK GPS, and soil moisture sensor. The robot will perform dynamic and efficient SOC assessments on the spot within the management zones identified by the UAV. Using the data collected, prescription maps will be created through a GIS platform, guiding the application of necessary soil treatments.

VRA spraying/Fertilization: In this process, prescription maps generated from the SOC assessments will be sent to farmers. The farmers will then provide detailed guidance on the required fertilizer volume per area. Variable Rate Application (VRA) equipment will be used to apply fertilizers accurately based on the prescription maps.

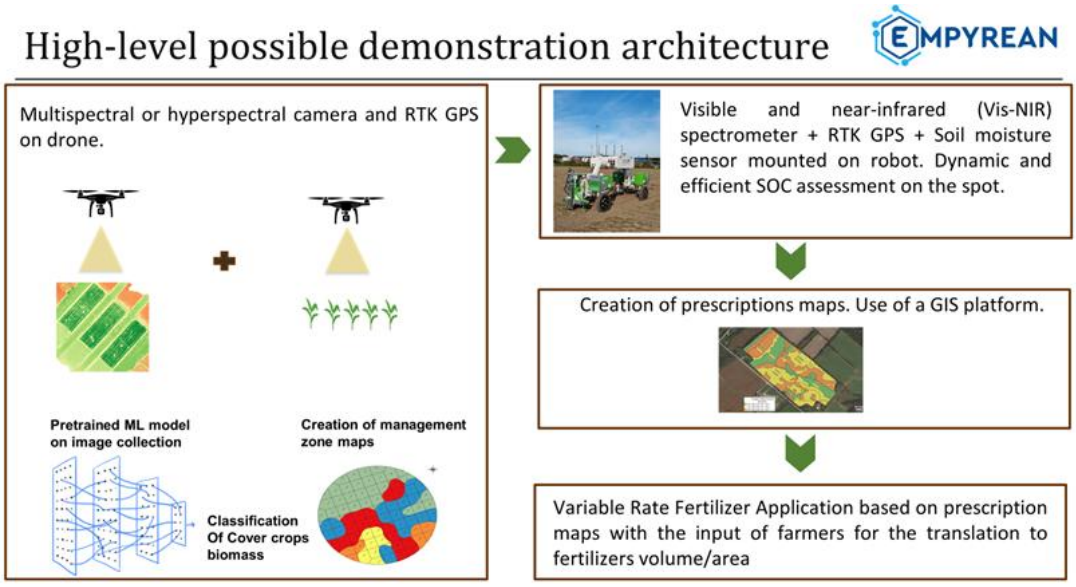

Figure 12: Indicative high-level architecture for the use case demonstration

4.3 5G-Enabled Vehicle-Assisted Services (UC3)

4.3.1 Overview

This use case aims to explore vehicle-assisted services in dynamic, latency-sensitive environments where computational tasks—such as real-time video, LIDAR, or telemetry processing must be offloaded to nearby edge infrastructure. The scenario focuses on how EMPYREAN's orchestration, telemetry, and security mechanisms can support mobility-aware service deployment and ensure continuous, trustworthy operation as vehicles move across 5G-connected domains.

Figure 13: GAIA Lab testbed at the University of Murcia

The experimentation will be carried out at the GAIA Lab testbed at the University of Murcia (UMU), which provides a 5G and edge computing environment suitable for testing orchestration and service continuity under vehicular mobility conditions.

4.3.2 Detailed Description

Modern connected and autonomous vehicles generate large volumes of data that require low-latency processing and adaptive orchestration to maintain performance and safety. Current cloud-centric approaches introduce delays and lack flexibility when vehicles change zones or network conditions vary.

This use case envisions a 5G-enabled orchestration ecosystem capable of deploying and migrating workloads dynamically between edge nodes based on vehicle location, resource availability, and network metrics. The EMPYREAN platform will provide the enabling technologies to achieve this vision, including:

- A Decision Engine for latency-aware workload placement and migration.
- A Telemetry Service for continuous monitoring of network and system parameters.
- A Cyber Threat Intelligence (CTI) Engine to anticipate and correlate security events.
- Edge-to-cloud storage (via Skyfolk's Edge Storage Gateway by Chocolate Cloud) for policy-based, distributed data management.

The Workflow Manager (RYAX) will serve as the coordination layer between these components, ensuring that deployment, telemetry, and data flows are handled autonomously and securely.

4.3.3 Current State - Future State with EMPYREAN

Current State:
Existing vehicular systems depend heavily on centralized cloud infrastructures. They typically lack mobility awareness and real-time orchestration, leading to performance degradation during handovers or connectivity fluctuations. Security mechanisms are often reactive rather than integrated into the orchestration process.

Future State with EMPYREAN:
Through EMPYREAN, this use case will evolve toward a fully integrated, cognitive orchestration framework capable of proactive decision-making and adaptive service migration.
Planned EMPYREAN contributions include:

- Dynamic Mobility-Aware Orchestration using the Decision Engine to select optimal edge nodes.
- Telemetry-Driven Anomaly Detection to identify performance or security anomalies in real time.

- Autonomous Security Mitigation deploying VNFs (e.g., filters or isolators) automatically upon detection of threats.
- Threat Intelligence Correlation integrating CTI data to improve detection accuracy.
- Integrated Edge-to-Cloud Storage enabling secure, policy-driven data management across the continuum.

At this stage, the infrastructure and architectural design are established, and the integration and implementation of EMPYREAN components within the GAIA Lab environment are planned for subsequent phases.

4.3.4 Challenges to be Addressed

- **Low-latency orchestration:** Achieving sub-second decision times for workload migration.
- **Service continuity:** Ensuring uninterrupted performance during 5G handovers.
- **Cybersecurity integration:** Embedding real-time detection and mitigation within the orchestration pipeline.
- **Data privacy and resilience:** Managing vehicular data securely across distributed edge-cloud environments.
- **Dynamic resource optimization:** Adapting to mobility, workload variations, and fluctuating network conditions.

4.3.5 Deployment Environment

The use case will be developed and validated in the GAIA Lab testbed at the University of Murcia, a controlled environment that supports advanced research in 5G networks, SDN/NFV, and edge computing.

GAIA Lab currently provides:

- A private 5G backbone connecting several campus sites (Luis Vives, ATICA, Bellas Artes, GAIA).
- Multiple distributed edge clusters based on OpenStack and Kubernetes.
- SDN-enabled networking for programmable routing and QoS control.
- NFV capabilities to deploy VNFs and CNFs dynamically.
- Integration points for EMPYREAN components such as the Orchestrator, Telemetry Service, Workflow Manager, and CTI Engine.

During the next phases, the EMPYREAN components will be deployed and tested in this environment, enabling real-world experimentation of service migration, telemetry analytics, and anomaly detection in mobility scenarios.

4.3.6 KPIs

The success of the **5G-Enabled Vehicle-Assisted Services** use case will be evaluated using several **Key Performance Indicators (KPIs)** that measure the platform's ability to deliver low-latency orchestration, reliable service continuity, and security responsiveness in mobile 5G environments.

These KPIs are designed to assess the effectiveness of EMPYREAN's orchestration, telemetry, and CTI mechanisms when applied to vehicular mobility scenarios. They will be refined and validated during later project stages, once the components are integrated into the GAIA Lab testbed.

No	Indicator	Success Criteria
1	Service offloading and orchestration decision latency	< 20 seconds
2	Security alert generation and mitigation initiation	< 2 seconds
3	False positive reduction using CTI-enhanced detection	≥ 30% improvement vs. telemetry-only baseline

1. **Service offloading and orchestration decision latency:** This KPI will measure the responsiveness of the orchestration layer in dynamically deploying or migrating vehicular services (e.g., virtual RSUs or inference engines) across edge nodes. The target is to achieve service redeployment within 20 seconds, maintaining operational continuity as vehicles move between 5G coverage zones.
2. **Security alert generation and mitigation initiation:** This KPI focuses on the system's ability to identify anomalies or security threats in near real-time and initiate automatic mitigation (e.g., deploying traffic filters or isolation VNFs). The expected response time for threat detection and mitigation initiation is below 2 seconds, ensuring timely protection of active services.
3. **False positive reduction using CTI-enhanced detection:** By combining telemetry data with contextual Cyber Threat Intelligence (CTI), EMPYREAN aims to improve detection precision. The goal is to achieve at least a 30% reduction in false positives compared to telemetry-only anomaly detection, demonstrating the added value of CTI correlation within the orchestration workflow.

4.3.7 Validation and testing

Validation activities will take place once the EMPYREAN components are deployed within GAIA Lab. The experiments will focus on:

- Evaluating orchestration and migration performance under mobility conditions.
- Testing telemetry-driven and CTI-assisted anomaly detection workflows.
- Assessing end-to-end latency, service continuity, and resource utilization.

- Validating security mechanisms such as DICE-based Secure Boot and DID/VC identity verification during virtual RSU (vRSU) migration.
- Collaborating with Kookmin University (South Korea) to replicate security scenarios and explore cross-continental applicability of EMPYREAN's orchestration and threat management solutions.

5 EMPYREAN Platform Components Definitions

5.1 EMPYREAN Ecosystem

The conventional way of dealing with (big) data generated and exchanged by IoT devices and robots is to continuously push them to centralized cloud computing infrastructures or utilize jointly on-device and on-premise/micro edge resources, deep and far edge computing nodes, along with the traditional federated cloud resources, thus forming an IoT-edge-cloud continuum. This approach can increase manifold the applications' performance and the efficiency of the infrastructures[142,143], overcoming the centralized systems' critical data collection, transmission, and processing bottlenecks.

However, most related works and projects treat this continuum as a monolithic pipeline where any resource can be accessed by anyone. While this is a scientifically viable assumption for research papers and a valid setup for developing higher-layer functionalities (e.g., orchestration, application deployment), it cannot realistically be applied in real-world scenarios. In reality, multiple organizations or individuals own and use IoT devices or robots with on-device local computing and storage resources and cloud computing infrastructures. As a result, edge resources (on-device, on-premise, near-edge, far-edge, fog, etc.) are fragmented, often underutilized, and generally disconnected from cloud resources and any market.

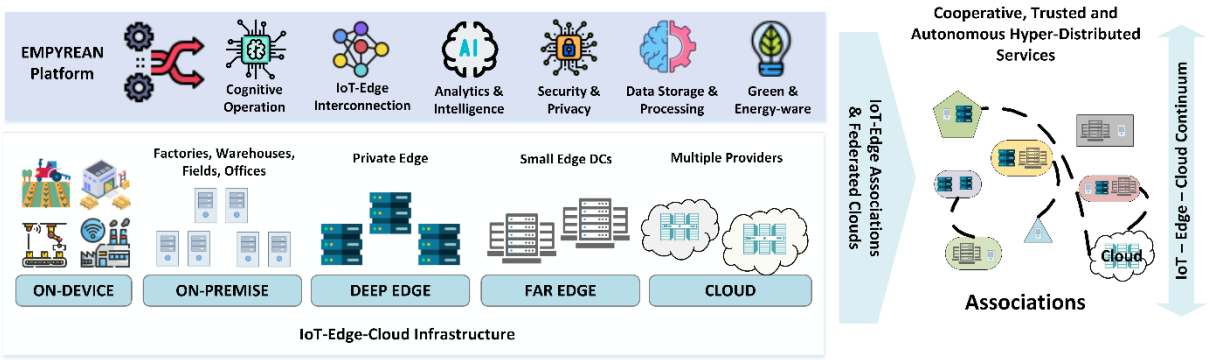

Figure 14: EMPYREAN overall concept and vision

EMPYREAN envisions a new paradigm for the continuum, introducing the concept of collaborative collectives of IoT devices, robots, and resources spanning from the edge to the cloud. EMPYREAN calls this the Association-based continuum (Figure 21), in the sense that multiple Associations (collaborative collectives of IoT devices, robots, and resources) operate in parallel in space and time and constitute the IoT-edge-cloud continuum. Each Association is composed of shared and aggregated edge computing and storage resources of various sizes and characteristics, encompassing both general-purpose and specialized units. These

[142] aioti.eu/wp-content/uploads/2020/10/IoT-and-Edge-Computing-Published.pdf

[143] atos.net/wp-content/uploads/2021/08/atos-2021-perspective-on-edge-computing-white-paper.pdf

Associations are dynamically formed and updated based on the resource owners' participation, while central cloud resources can be utilized when and if needed. This approach promises to enhance the flexibility, efficiency, and collaboration within the IoT-edge-cloud ecosystem.

In particular, EMPYREAN's Associations notion:

- **Supports Collaborative Continuum**: Empower organizations to build a collaborative IoT-edge-cloud continuum utilizing self-owned resources.
- **Enables Scalability**: Allows an Association to scale by involving multiple users/organizations and facilitating resources sharing between them.
- **Abstracts Complexity:** Simplifies the complexity and dynamicity of the underlying infrastructures that usually belong to different administrative domains.
- **Maximizes Resource Utilization**: Overcomes the isolation and underutilization of edge resources.
- **Promotes Self-Sufficiency**: Advocates for a self-sufficient IoT-edge continuum, acknowledging that the cloud may not always be available or its use may be prohibited for various reasons.
- **Facilitates Inter-Association Cooperation**: Supports cooperation between different Associations, enabling the use of resources under stricter access and security rules compared to resources belonging to an Association.

The core idea of Associations, those collaborative collectives, is not entirely new and finds parallels in various domains and society in general. For instance, energy collectives have been proposed and investigated within energy grids. In related work[144], the concept of energy collectives is introduced as a community-based electricity market structure. Moreover, resource aggregation has been proposed for energy grids under the vision of micro-grids, as demonstrated in the VIMSEN project[145]. Micro-grids, composed of distributed small energy producers, aggregate their resources into larger associations (often called Virtual Power Plants) to optimize collective benefits. In this case, small energy producers resemble distributed edge resources, while large energy producers are akin to cloud providers. Additionally, associations are prevalent in the farming and agricultural domain, where groups of farmers collaborate on activities related to the marketing or selling of agricultural products, as well as the growing, harvesting, processing, and packing of these products, and sharing equipment, fertilizers, and other resources.

EMPYREAN's Association does not propose a greenfield-like change in the domain but rather a new and viable way to organize existing and future resources in a brownfield manner. This

[144]Moret, F., & Pinson, P. (2019). Energy Collectives: a Community and Fairness based Approach to Future Electricity Markets. IEEE Transactions on Power Systems, 34(5), 3994-4004. https://doi.org/10.1109/TPWRS.2018.2808961

[145] D. J. Vergados, I. Mamounakis, P. Makris, E. Varvarigos, "Prosumer Clustering into Virtual Microgrids for Cost Reduction in Renewable Energy Trading Markets", Elsevier Sustainable Energy, Grids and Networks (SEGAN), Vol. 7, pp. 90-103, September 2016, https://www.sciencedirect.com/science/article/pii/S2352467716300297.

approach constructs the IoT-edge-cloud continuum as an Association-based continuum. To this end, EMPYREAN's software components, platform mechanisms, and decision-making algorithms will be fully reusable and adaptable, enhancing the flexibility and scalability of the EMPYREAN platform.

Figure 22 illustrates the EMPYREAN ecosystem, highlighting the main stakeholders, their roles, and interactions. The envisioned ecosystem promotes the composability of infrastructures and services across the IoT-edge-cloud continuum. Within this framework, Associations facilitate the collaborative operation and management of virtual executing environments by pooling computational, storage, networking, and other infrastructure and service resources. The key stakeholders are: (i) infrastructure providers, (ii) service providers, (iii) application developers, (iv) EMPYREAN customers, and (iv) end users.

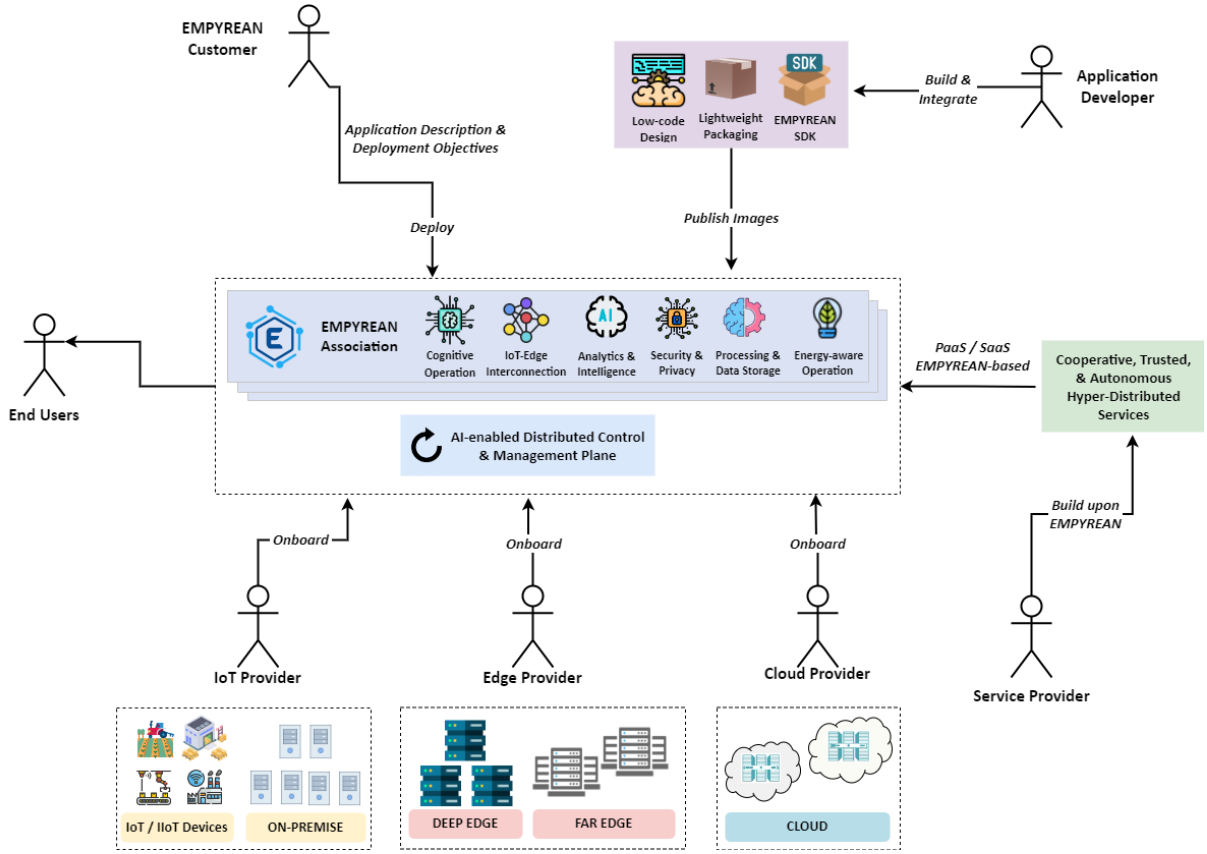

Figure 15: EMPYREAN ecosystem, key stakeholders and interactions

The *infrastructure providers* include:

- IoT providers that offer IoT infrastructures across the continuum, where data is produced and service requests are generated, with IoT and Industry IoT (IIoT) devices and on-premise low-capacity edge resources. Through the EMPYREAN platform, these providers can offer their infrastructure to multiple vertical applications.
- Edge providers that offer deep and far edge computing and storage resources, close and further from the end users/devices. These resources, coupled with hardware

acceleration capabilities suitable for AI/ML workloads, will be utilized for time-critical and resource-demanding workloads.

- Cloud providers with centralized computational and storage resources whose seamless inclusion in the Association-based continuum can increase robustness and reduce cost. These resources will be used for computationally-intensive and latency-tolerant workloads, as well as long-term data storage and replication.

The *service providers* include domain-specific and generic platforms built upon the Association-based continuum. These platforms will enable optimal deployment and autonomous adaptations of continuum-native applications and extreme-scale distributed AI/ML workflows over heterogeneous and trusted resources. Both infrastructure and service providers will leverage EMPYREAN's AI-enabled management mechanisms and trustworthy techniques to automate and optimize internal operations. This will facilitate collaboration for seamless deployment and distributed data processing across the entire continuum.

The *application developer* creates hyper-distributed continuum-native applications. This involves developing new functionalities or enhancing existing ones to fully leverage EMPYREAN's cutting-edge technological advancements. Depending on the application's complexity, the developer may also serve as the integrator. This stakeholder utilizes EMPYREAN's workflow-based design, lightweight environment packaging, and low-code description to design the application, package its logic, and define deployment objectives in a generic, infrastructure-agnostic manner.

An *EMPYREAN customer* utilizes the platform to deploy these applications across the Association-based continuum, benefiting from trustworthy, autonomous, scalable, and collaborative data processing capabilities. Finally, the *end-users* are those who interact with and use the hyper-distributed applications deployed by EMPYREAN customers on the platform.

The EMPYREAN ecosystem is designed for flexibility, allowing stakeholders to take on multiple roles. For example, an organization can act as both an infrastructure provider and an EMPYREAN customer. In this dual role, it can contribute a portion of its infrastructure resources to the EMPYREAN platform via an Association, making them available to other EMPYREAN customers. At the same time, as an EMPYREAN customer, the organization can utilize the platform's decentralized intelligence and application development and deployment solutions to enhance its own application performance.

EMPYREAN serves as a bridge between infrastructure and service providers (the supply side) and application developers and end users (the demand side) who require high-performing, low-latency, hyper-distributed applications. EMPYREAN aims to achieve optimal balance between maximizing resource utilization and collaboration, thereby generating revenue for the supply side, while ensuring the highest quality of service and experience for the demand side.

5.2 Components Description

This section provides a high-level overview of the key components of the EMPYREAN platform, showcasing innovative developments that facilitate a collaborative cognitive continuum. By integrating intelligence and automation, these components enhance data processing efficiency. We provide a high-level description and categorization of each component, along with a brief overview of their dependencies, interfaces, involved partners, and relation to the project use cases. These preliminary definitions aim to elicit functional and non-functional requirements (Section 6). A more detailed technical presentation, including the components' interactions and interfaces, will be available in deliverable D2.2 "Initial release of EMPYREAN architecture" (M7).

5.2.1 Privacy and Security Manager

- **Stakeholders:** End Users, Application Developers, Infrastructure Owners
- **High-level category:** Security and Privacy Management
- **High-level description:** This component focuses on cybersecurity, privacy-preserving techniques, and identity management using decentralized identifiers (DIDs). It handles authorization with verifiable credentials (VC) and includes verifiable presentations with zero-knowledge proofs (ZKPs) and selective disclosure. The verification is performed through Distributed Ledger Technologies (DLTs), and smart contracts are used to retrieve and store DIDs.
- **Dependencies**: p-ABC Library, Hyperledger Fabric
- **Interfaces**:
  - Interfaces with DLTs for verification
  - Interfaces with smart contract systems for storage and retrieval of DIDs
- **Relation with use case**:
  - Ensures secure and private identity management and data verification within the EMPYREAN platform.
- **Involved Partner:** UMU
- **Enabler:** EN_1
- **Priority:** High
- **KPIs:**
  - T3.1
  - T3.2
  - T3.3
- **Related Requirements:**
  - F_ST.1, F_ST.2

5.2.2 P-ABC

- **Stakeholders:** Application Developers, Security Experts
- **High-level category:** Security and Cryptography Library
- **High-level description:** This component provides a distributed privacy-preserving attribute-based credential system based on PSMS (Pointcheval–Sanders Multi-Signatures). It uses a wrapper library for elliptic curve (EC) arithmetic to support these functionalities.
- **Dependencies:** self-contained library.
- **Interfaces:** C Library compiled as a dependency, Provides APIs for cryptographic operations related to attribute-based credentials.
- **Relation with use case:** Supports the privacy and security requirements of the EMPYREAN platform by enabling attribute-based credential management.
- **Involved Partner:** UMU
- **Enabler:** EN_1
- **Priority:** Medium
- **KPIs:**
  - T3.1
  - T3.2
  - T3.3
- **Related Requirements**
  - F_ST.1

5.2.3 Telemetry Service

- **Stakeholders:** System Administrators, Infrastructure Owners, Security Analysts
- **High-level category:** Monitoring and Observability
- **High-level description:** The Telemetry Service handles observability and telemetry for the infrastructure, allowing system administrators to understand system behavior, troubleshoot problems, and monitor performance metrics like CPU, memory, storage, and network traffic in real-time. It includes alerting, dashboarding, and data transformation capabilities to optimize resource utilization and enhance security.
- **Dependencies:** -
- **Interfaces:**
  - Provides APIs for data collection and telemetry
  - Integrates with monitoring tools and dashboards
- **Relation with use case:** Provides real-time monitoring and telemetry data to support the secure and efficient operation of the EMPYREAN platform.
- **Involved Partner:** UMU

- **Enabler**: EN_2
- **Priority**: High
- **KPIs:**
  - T2.1
  - T1.4
- **Related Requirements:**
  - F_GR.6

5.2.4 Software-defined RDMA-based Unified Transport Service based on FlexDriver

- **Stakeholders:** Application Developers, Infrastructure Providers
- **High-level category:** Transport Service/Interconnect
- **High-level description:** It provides reliable RDMA-based transport service that leverages ring-buffer lock free structures as I/O interfaces to materialize the Proactor software design pattern that decouples requests from completion events. The approach offers the ability to integrate remote I/O operations into large computational pipelines (e.g. like AI training) and optimally overlap computation with network I/O.
- **Dependencies:** Requires the hardware platforms to feature RDMA-capable Network Interface Cards (NICs) and in case of hardware accelerators, specific FPGA integrated with RDMA NIC on the PCI-e bus.
- **Interfaces:**
  - Custom pub/sub interface for software applications. Stream interface for H/w accelerators with separate H/w pipelines for sending and receiving data.
- **Relation with use case:** Not directly related to a specific use case, primarily will be integrated into NUBIS containers as a general I/O service that any use case can take advantage of.
- **Involved Partner:** NVIDIA, NUBIS
- **Enabler:** EN_3
- **Priority:** High
- **KPIs:**
  - T4.1
  - T4.2
- **Related Requirements:**
  - F_ASSOC.5, F_ASSOC.6

5.2.5 Decentralized and Distributed Communication Layer

- **Stakeholders:** Infrastructure owners, Network administrators, Application Developers.
- **High-level category:** (i.e. Orchestration, Storage, App development, etc): real-time communication middleware, decentralized and distributed storage support, pub/sub mechanisms, distributed queries support.
- **High Level description:** The decentralized and distributed communication layer offers functionalities such as data dispatcher, which implements a networking layer capable of running above a Data Link, Network or Transport Layer. This provides primitives for efficient pub/sub and distributed queries. It supports fragmentation and ordered reliable delivery of messages. The initial design of the communication layer has been contributed and integrated into the Eclipse Zenoh[146] open source project. The EMPYREAN platform decentralized and distributed data communication component and Eclipse Zenoh are therefore used interchangeably on this deliverable since Zenoh is an actual implementation of it.
- **Dependencies:** This component has been developed in RUST, the exhaustive list of dependencies can be found at the cargo.lock file which is located at the root of the of the Eclipse zenoh source code.
- **Interfaces:** Custom pub/sub interface for software applications. Additionally, it supports distributed queries based on data named networking, thus it is location transparent, and supports geo-distributed storages. Data is organized as a set of **Resources**, where a resource is made up of a *key* and a *value*. Other important concept are key expressions, such as *robot/sensor/temp*, *robot/sensor/**, *robot/***, etc. The above key expression denotes set of keys, while the * and ** are wildcards representing respectively (1) an arbitrary string of characters, with the exclusion of the / separator, and (2) an arbitrary sequence of characters including separators.
- **Relation with use case:** Used in the tooling robotics use case lead by Tractonomy for robot to everything (R2X) communication.
- **Involved Partner:** ZSCALE
- **Enabler:** EN_4
- **Priority:** High
- **KPIs:**

Real-time data exchange and retrieval is complex and requires high performance both in terms of latency and throughput. High-performance needs first and foremost be provided by the underlying infrastructure (e.g. 4G/5G) and ultimately by the application. While the infrastructure oversees distributing the data, applications are in charge of processing the data, which may significantly impact the end-to-end performance of the different EMPYREAN use case.

[146] https://github.com/eclipse-zenoh/zenoh

- **Throughput--**: the application should be able to exploit no less than 80% of the available throughput provided by the 4G/5G link, e.g. in the order of hundreds of Mbps/tens of Gbps. To measure the throughput we expect to benchmark Zenoh throughput against standard throughput testing suites like iperf.
- **Latency**: the application should not perceive an increment in latency greater than 20% compared to what is provided by the infrastructure. To measure the latency we expect to benchmark Zenoh latency against standard latency testing suits like ping.

Related Requirements:

- In the context of mobile robots or vehicles, the devices should be able to change the point of attachment for their communication as fast as possible to minimize communication disruption.

5.2.6 Edge Storage Gateway

- **Stakeholders:** Application developers, Infrastructure owners
- **High-level category:** Storage
- **High-level description:** This component provides an access point to the EMPYREAN storage service through an industry-standard S3 interface. By defining a storage policy, users can precisely define the storage locations where their data should be distributed, the level of required redundancy, encryption, and compression scheme. Hybrid policies that use both cloud and edge locations are supported. Internally, it communicates with cloud storage locations, Edge Storage resources and the SkyFlok.com backend.
- **Dependencies:**
  - Edge Storage Gateway
- **Interfaces:**
  - S3 interface for data access
  - Storage policy API for defining where and how files are stored
  - REST API for listing available storage locations (cloud and edge)
- **Relation with use case:** the component can be utilized by any and all use cases that require object storage at the edge or in the cloud.
- **Involved Partner:** Chocolate Cloud
- **Enabler:** EN_5
- **Priority:** High
- **KPIs:**
  - T3.4
  - T3.5
- **Related Requirements:**
  - F_ASSOC.1,F_ASSOC.4, F_ASSOC.5
  - F_DI.4

- F_ST.1
- F_DCM.1
- NF_GR.1, NF_GR.4, NF_GR.5, NF_GR.6

5.2.7 Edge Storage

- **Stakeholders:** Application developers, Infrastructure owners
- **High-level category:** Storage
- **High-level description:** This component provides a common abstraction of storage resources located at the edge. Under the hood, a containerized version of Min.io is deployed. Any storage resource (e.g. local file system, NFS, etc.) that can be attached to the container as a volume can thus be exposed through a standard S3 interface, making it accessible for use by the Edge Storage Gateway.
- **Dependencies:**
  - Edge Storage Gateway
- **Interfaces:**
  - S3 interface for data access
  - Prometheus-compatible monitoring interface
- **Relation with use case:** the component can be utilized by any and all use cases that require storage at the edge.
- **Involved Partner:** Chocolate Cloud
- **Enabler:** EN_5
- **Priority:** High
- **KPIs:**
  - T3.4
  - T3.5
- **Related Requirements:**
  - Edge Storage Gateway requirements
  - F_GR.5, F_GR.6

5.2.8 IoT Query Engine

- **Stakeholders:** Application developers, Infrastructure owners
- **High-level category:** Storage
- **High-level description:** This component allows users to store and access IoT time series data. The novelty of the approach lies in the use of erasure coding. When accessing data during a query, only parts of the stored coded fragments are retrieved (compared to the entire fragment in a naive conventional solution). Depending on the

requirements of the use cases, it might be developed as a stand-alone component or be an integrated subcomponent of the Edge Storage Gateway.

- **Dependencies:**
  - Edge Storage Gateway
- **Interfaces:**
  - S3 interface for data access
  - Prometheus-compatible monitoring interface
- **Relation with use case:** The component can be utilized by any and all use cases that store and later run analytics workloads on time-series data. The benefits of the solutions will be most apparent in scenarios with large data volumes with many dimensions and queries that only need to evaluate a small subset of the dimensions.
- **Involved Partner:** Chocolate Cloud
- **Enabler:** EN_5
- **Priority:** High
- **KPIs:**
  - T4.4
  - T4.5
- **Related Requirements**:
  - Most of the requirements described for the Edge Storage Gateway
  - F_DCM.2

5.2.9 RYAX Workflow Engine

- **Stakeholders:** End Users, Application Developers, Infrastructure owners
- **High-level category:** Orchestration, Application development
- **High-level description:** This component involves the workflow management brought by RYAX engine**.** RYAX open-source workflow engine enables the design, deployment and monitoring of workflows of data analytics upon Cloud, Edge, HPC infrastructures. It makes use of Kubernetes orchestration and it provides a custom hybrid serverless based runtime environment for the deployment of components upon the related hybrid infrastructure. In the context of EMYREAN, RYAX workflow engine will be enhanced to support multi-site deployments the support of Zenoflow dataflow programming framework and the concepts of associations and aggregators.
- **Dependencies:** Service Orchestrator, Telemetry Service, AI-enabled workload autoscaling, Decisions Engine, Registry, Aggregator, NIX-based environment packaging.
- **Interfaces:** API, Web and CLI

- **Relation with use case:** It will enable users to design the applications in all use cases through workflows along with the deployment and monitoring of their executions upon the related computing continuum.
- **Involved Partner:** RYAX
- **Enabler:** EN_6
- **Priority:** High
- **Related Requirements:**
  - F_GR_1, F_ASSOC_1
  - F_DI(1-9), F_SO.15, F_SO.14, F_SO.10, F_SO.9, F_SO.8, F_SO.7, F_SO.6, F_SO.5, F_SO.4, F_SO.3, F_SO.2, F_SO.1
  - F_IPDR.3, F_IPDR.2
  - NF_GR.1, NF_GR.2, NF_GR.3, NF_GR.4, NF_GR.5, NF_GR.8

5.2.10 AI-enabled Workload Autoscaling

- **Stakeholders:** End Users, Infrastructure owners
- **High-level category:** Orchestration
- **High-level description:** This component will provide an AI-enabled workload autoscaling mechanism, which will be based upon Kubernetes orchestrator enhanced with AI/ML techniques for intelligent resource requests and limits allocation. Specifically, we will use AI/ML to perform optimal workload autoscaling by setting the most adequate resource limits configuration and performing dynamic adaptation based on historical data of previous executions. Hence, the component will provide enhancements on available Kubernetes autoscaling mechanisms.
- **Dependencies:** Telemetry Service, Service Orchestration
- **Interfaces:** Kubernetes API
- **Relation with use case:** It will enable infrastructure owners and platform service provides to improve the system utilization through a better utilization of available resources.
- **Involved Partner:** RYAX
- **Enabler:** EN_7
- **Priority:** High
- **Related Requirements:**
  - F_GR.3, F_GR.7
  - F_DI.7, F_DI.8, F_DI.9

5.2.11 CTI Analysis Module

- **Stakeholders:** Infrastructure owners, Security Experts
- **High-level category:** Security, Orchestration
- **High-level description:** A module that provides information about past Cyber Threat Intelligence (CTI) events observed around the world. It may serve as an external source to quantify the risk of different systems.
- **Dependencies:**
  - External access to CTI sources
- **Interfaces:**
  - GUI for security experts
  - API
- **Relation with use case**: It will enable access to external CTI data and provide critical feedback to EMPYREAN orchestration and service assurance mechanisms to trigger updates in Associations and migrate deployed workloads
- **Involved Partner:** NEC
- **Enabler:** EN_8
- **Priority:** Medium
- **Related Requirements:**
  - F_ASSOC.4
  - F_ST.4

5.2.12 Decision Engine

- **Stakeholders**: Infrastructure Owners, End Users
- **High-level category**: Orchestration
- **High-level description**: It provides the implementation of the developed multi-objective optimization and orchestration algorithms. Specific task placement policies involving energy consumption and others related to cold-start delays optimizations are also included. It receives execution requests from the Service Orchestrator and also interacts with the Telemetry Service to retrieve the required information.
- **Dependencies**: Service Orchestrator, Telemetry Service
- **Interfaces**:
  - REST
  - Asynchronous message-based
- **Relation with use case**: It will provide the required orchestration decisions to EMPYREAN Service Orchestrator to assign and re-optimize the UC applications' workloads.

- **Involved Partner**: ICCS, RYAX
- **Enabler:** EN_9, EN_17
- **Priority**: High
- **Related Requirements**:
  - F_GR.6
  - F_DI.1, F_DI.2, F_DI.4, F_DI.5, F_DI.6
  - F_SO.9

5.2.13 Analytics Engine

- **Stakeholders**: Infrastructure Owners, Application Developers, End Users
- **High-level category**: Service Assurance
- **High-level description**: It implements the service assurance mechanisms within the EMPYREAN platform to detect issues with the operation of the infrastructure resources and Associations along with the performance of the deployed applications. It will analyze the collected telemetry data in order to trigger pro-actively and re-actively dynamic adjustments.
- **Dependencies**: Service Orchestrator, Telemetry Service, Data Distributor
- **Interfaces**:
  - REST
  - Asynchronous message-based
- **Relation with use case**: It will ensure that the applications perform as intended, while it will dynamically trigger the necessary adjustments if the current deployments do not comply with the requested SLA guarantees.
- **Involved Partner**: ICCS, RYAX, UMU, ZSCALE
- **Enabler:** EN_2
- **Priority**: High
- **Related Requirements**:
  - F_GR.6
  - F_ASSOC.9
  - F_DI.6, F_DI.7, F_DI.8
  - F_SO.5

5.2.14 EMPYREAN Orchestrator and Controller

- **Stakeholders**: Infrastructure Owners, Service Providers, End Users
- **High-level category**: Orchestration and Deployment
- **High-level description**: The EMPYREAN Orchestrator ensures efficient service orchestration and resource management in the disaggregated and heterogeneous EMPYREAN infrastructure. It initiates the application deployment and automatically coordinates the necessary supplemental actions (e.g., transfer of required data). The EMPYREAN Controller abstracts the interaction with the specific edge and cloud orchestration mechanisms at each EMPYREAN Association.
- **Dependencies**: Decision Engine, Telemetry Service, Analytics Engine, Secure Storage Service
- **Interfaces**: REST interface
- **Relation with use case**: They will prepare, coordinate, and manage the application deployment at the selected individual edge and cloud platforms.
- **Involved Partner**: ICCS, RYAX, NUBIS
- **Enabler:** EN_9
- **Priority**: High
- **Related Requirements**:
    - F_GR.1, F_GR.3, F_GR.4
    - F_ASSOC.4, F_ASSOC.7
    - F_DI.3, F_DI.4, F_DI.5
    - F_SO.3, F_SO.4, F_SO.7

5.2.15 Telemetry Engine

- **Stakeholders**: Infrastructure Owners, Application Developers, End Users
- **High-level category**: Infrastructure and Application Monitoring
- **High-level description**: The main component in EMPYREAN distributed telemetry infrastructure. It maintains a global view of the state of the infrastructure resources and the deployed applications. It coordinates the operation of a specific set of Monitoring Probes that are responsible for monitoring the resources and the deployed applications. Moreover, it provides historical telemetry data to enable their cognitive orchestration and re-optimization.
- **Dependencies**: Telemetry Service, Service Orchestrator, Decision Engine, Analytics Engine, Monitoring Probes
- **Interfaces**: REST interface, Python API
- **Relation with use case**: It will provide the required information for the EMPYREAN orchestration and decision-making mechanisms. Thus, it will contribute to UC

applications' cognitive orchestration and deployment in the EMPYREAN platform.

- **Involved Partner**: ICCS, UMU, NEC, RYAX, NUBIS, CC
- **Enabler:** EN_2
- **Priority**: High
- **Related Requirements**:
  - F_GR.3, F_GR.6
  - F_ASSOC.8, F_ASSOC.9, F_ASSOC.10
  - F_DI.6

5.2.16 EMPYREAN Registry

- **Stakeholders**: Infrastructure owners, Application Developers, End Users
- **High-level category**: Infrastructure Management
- **High-level description**: It manages the registration of IoT devices, edge, and cloud resources in Associations. It also abstracts to the Workflow Manager the interaction with the available Associations. The EMPYREAN registry will keep track of the available Associations, services, and container images, the mapping of the infrastructure resources to Associations, and the relation between users and Associations.
- **Dependencies**: EMPYREAN Aggregator, Workflow Manager, Lightweight Application Packaging, API Gateway
- **Interfaces**:
  - REST interface
  - Asynchronous interface for notifications
- **Relation with use case**: It will facilitate the seamless deployment of EMPYREAN use case applications across an Association-based IoT-edge-cloud continuum. It will handle along with the Workflow Manager the initial steps of the applications' lifecycle workflow within the EMPYREAN platform.
- **Involved Partner**: ICCS, RYAX, NUBIS, UMU
- **Enabler:** EN_10
- **Priority**: High
- ***Related Requirements***:
  - F_GR.1
  - F_ASSOC.8
  - F_SO.5

5.2.17 EMPYREAN Aggregator

- **Stakeholders**: End Users, Application Developers, Infrastructure owners
- **High-level category**: Orchestration and Deployment
- **High-level description**: The EMPYREAN Aggregator manages and coordinates the operation of an EMPYREAN Association. Each Aggregator includes several core services that provide the required intelligence and orchestration logic to operate an Association, deploy workloads, and manage data access and storage. An Aggregator orchestrates its own Associations that include separate or shared computational and storage resources.
- **Dependencies**: EMPYREAN Registry, Service Orchestrator, Security and Trust Manager, Data Distributor, Decision Engine, Telemetry Service, Edge Storage Gateway
- **Interfaces**: REST interface, Asynchronous interface for notifications
- **Relation with use case**: It will provide and manage the Association-based IoT-edge-cloud continuum to ensure the performance, security and energy efficiency for the use cases workloads. Additionally, it will abstract the inter-Association interactions and facilitate their collaborative operation.
- **Involved Partner**: ICCS, UMU, CC, ZSCALE, RYAX, NUBIS
- **Enabler:** EN_11
- **Priority**: High
- **Related Requirements**:
  - F_GR.2
  - F_ASSOC.1, F_ASSOC.4, F_ASSOC.7, F_ASSOC.8, F_ASSOC.9, F_ASSOC.10

5.2.18 vAccel

- **Stakeholders**: Application Developers, Infrastructure owners
- **High-level category**: Orchestration and Deployment
- **High-level description**: vAccel is an open-source framework designed to enable flexible execution by mapping hardware-accelerate-able workloads to relevant hardware functions, thus decoupling applications from hardware-specific code. It aims at enhancing security by ensuring that consecutive executions on a hardware-accelerated platform do not leak sensitive data. This framework is part of the EMPYREAN project's initiative to facilitate the development and deployment of compute-intensive functions across IoT devices and edge nodes, leveraging the concept of remote hardware accelerators. IoT devices can use the vAccel API to request compute-intensive tasks to be executed by an available neighboring node within the Association, integrating with established open-source solutions at the systems level (e.g., Kubernetes, K3s, OpenFaaS) and including their high-level APIs in the EMPYREAN SDK to simplify application development and deployment.

- **Dependencies**: EMPYREAN SDK, Open-source solutions (e.g., Kubernetes, K3s, OpenFaaS), Flexible Hardware-Accelerated Execution, Offload Acceleration to Nearby Devices
- **Interfaces**: API for triggering vAccel functions from the user.
- **Relation with use case**: UC1, UC3
- **Involved Partner**: NUBIS
- **Enabler:** EN_12
- **Priority**: High
- **Related Requirements:**
  - F_GR.5
  - F_SO.11

5.2.19 Application Builder for Unikernels

- **Stakeholders**: Application Developers, Infrastructure owners
- **High-level category**: Orchestration and Deployment
- **High-level description**: This component is designed to address the deployment of applications in cloud-native environments using unikernels. It aims to tackle two major challenges associated with unikernels: (i) simplifying the building and deployment process, and (ii) minimizing the engineering overhead to resolve external software dependencies.
- **Dependencies**: Simplification of the building and deployment process for unikernels, Minimization of engineering overhead to resolve external software dependencies.
- **Interfaces**:
  - inputs: Application description & source/binary repository of the application
  - outputs: binary artifact (unikernel, bootable using a hypervisor, or on bare metal)
- **Relation with use case**: UC2, UC3
- **Involved Partner**: NUBIS
- **Enabler:** EN_13
- **Priority**: High
- **KPIs**:
  - T5.1
- **Related Requirements:**
  - F_SO.14

5.2.20 Application Packaging

- **Stakeholders**: Application Developers, Service Providers
- **High-level category**: Orchestration and Deployment
- **High level description**: This component is designed to streamline the application packaging process across diverse computing environments, focusing on creating OCI-compatible container images. This tool aims to bundle binary artifacts along with their descriptors into OCI container images, facilitating deployment across EMPYREAN's supported execution modes, including containers, sandboxed containers, WebAssembly (WASM), unikernels, and binary blobs for IoT devices. This development is crucial for EMPYREAN's overarching goal of enabling seamless application deployment and execution across heterogeneous hardware architectures and environments, enhancing interoperability, and ensuring efficient, cloud-native deployment methodologies
- **Dependencies**: Automation and orchestration tools. Common schemas for describing deployment objectives and end user preferences.
- **Interfaces**:
    - inputs:
        - Application description (Dockerfile-like)
        - source/binary repository of the application (combined with Application Builder component), or
        - unikernel binary
    - outputs: OCI artifact bootable using component WP4.Task3.[3]
- **Relation with use case**: All
- **Involved Partner**: NUBIS
- **Enabler:** EN_14
- **Priority**:
- **KPIs**:
    - T5.2
    - T5.3
- **Related Requirements:**
    - F_SO.6, F_SO.15

5.2.21 Container Runtime

- **Stakeholders**: Infrastructure Owners
- **High-level category**: Orchestration and Deployment
- **High-level description**: The component within the EMPYREAN project aims at facilitating the deployment of applications across various execution environments, including unikernels and IoT devices. This component is based on urunc, a runtime capable of spawning unikernels and seamlessly integrating them with generic

container runtimes compatible with Kubernetes and serverless architectures.

This component allows for the execution of applications built with the "Application builder" component within the existing container orchestration ecosystems, providing the benefits of diverse building systems (e.g., unikernels for improved security and performance) while maintaining compatibility with widespread deployment models.

- **Dependencies**: Common schemas for describing deployment objectives and end user preferences. Comprehensive monitoring and analytics mechanisms. Automated and data-driven orchestration tools. Interoperable description of deployment policies.
- **Interfaces**:
  - inputs: OCI artifact, metadata
  - outputs: successful execution of the binary artifact
- **Relation with use case**: All
- **Involved Partner**: NUBIS
- **Enabler:** EN_15
- **Related Requirements:**
  - F_SO.3**,** F_SO.4**,** F_SO.13**,** F_SO.15

5.2.22 Secure execution environment

- **Stakeholders**: Infrastructure Owners
- **High-level category**: Orchestration and Deployment
- **High-level description**: This component is focused on establishing a secure and trusted execution environment across the IoT-edge-cloud continuum. This environment based on unikernels will support secure and measured boot mechanisms that are tightly coupled with the systems layer. This approach will ensure that applications can be deployed with varying levels of security and trustworthiness across different hardware, enabling scalable and transparent operation from the micro deep edge to far edge and to cloud environments without altering the deployment descriptor or application logic. Furthermore, it will address the trade-off between flexible workload deployment and the single-tenant use of computing resources, particularly in energy and resource-constrained edge platforms.
- **Dependencies**: Secure and measured boot mechanism, Systems layer integration, Support for varying levels of security and trustworthiness, Deployment across IoT-edge-cloud continuum, Use of unikernels for security, Consistent deployment without altering logic, Balance flexible deployment with single-tenant use, Energy and resource constraints consideration.
- **Relation with use case**: UC2, UC3
- **Involved Partner**: NUBIS
- **Enabler:** EN_16
- **Priority**: High

- **Related Requirements:**
  - F_ASSOC.3
  - F_SO.7**,** F_SO.14
  - F_ST.6

5.2.23 NIX-based Environment Packaging

- **Stakeholders**: Application Developers
- **High level category**: Application development
- **High level description**: This is a tool used in the context of Ryax workflow management and it involves the mechanism of performing multi-arch and polyglot environment packaging to build the components to be used within the workflows. This mechanism will be enhanced to support Web Assembly and unikernels.
- **Dependencies**: RYAX workflow engine, Application Packaging, Application Builder for unikernels
- **Interfaces**: Ryax workflow engine APIs, WebUI, CLI along with NIX related CLI
- **Relation with use case**: All use cases
- **Involved Partner**: RYAX
- **Enabler:** EN_14
- **Priority**: High
- **Related Requirements:**
  - F_SO.13, F_SO.14, F_SO.15
  - F_IPDR.4

5.3 Technical KPIs

The following table provides a regrouping of the different KPIs defined in the DoA of EMPYREAN upon which the different components described in the previous section are aligned. Besides the number and the name of the KPI it also provides a success criteria which will allow us to track its usage through the project.

Table 3: EMPYREAN Technical KPIs

ID	Summary	Description	Success Criteria	Baseline	Objective
T1.1	Reduce cloud and increase edge utilization via workload balancing optimization.	The use of Associations will enable increased utilization of edge resources compared to scenarios where limited-capacity edge nodes operate and cloud resources are required to handle workload-intensive tasks.	50% reduction in core cloud	Traditional centralized cloud related services	Obj.1
T1.2	Increase reliability in the edge.	Edge resources within an Association will collaboratively execute workloads, achieving higher reliability compared to scenarios where edge nodes operate independently.	>50% increase compared to SotA	Non-federated computing envrironments	Obj.1
T1.3	Increase statistical multiplexing gains through associations.	By dynamically pooling and sharing resources across multiple edge nodes within an Association, workload fluctuations can be balanced more efficiently, leading to higher statistical multiplexing gains compared to isolated edge or cloud deployments.	x2 compared to standard execution	Non-federated computing envrironments	Obj.1
T1.4	Provide low and predictable latency for hyper-distributed applications.	Through intelligent workload placement and coordination within Associations, applications spanning multiple edge and cloud layers will experience consistently low and stable latency, outperforming conventional, cloud-centric execution models.	<1 ms for delay-sensitive apps	Traditional centralized cloud related services	Obj.1

T2.1	Improve overall performance compared to SotA.	EMPYREAN will outperform centralized and static orchestration mechanisms through AI-driven adaptive orchestration across the Association-based continuum.	by 40%	Centralized and static orchestration mechanisms	Obj.2
T2.2	Reduce energy consumption on Associations compared to standard execution.	EMPYREAN will reduce energy consumption within Associations compared to static and non-coordinated execution by leveraging AI-driven adaptive and cooperative orchestration across the Association-based continuum.	>25%	Static and non-coordinated execution	Obj.2
T2.3	React fast to rapid changes in computational and data demands so as to maximize the number of demands served.	React rapidly to dynamic computational and data demands within Associations, where cooperative edge resources enable intelligent autoscaling and adaptive task redistribution, outperforming non-federated solutions in maximizing the number of served demands.	between x2 and x10 increase	Non-federated computing envrironments	Obj.2
T2.4	Boost AI-driven decision-making accuracy.	EMPYREAN will boost AI-driven decision-making accuracy by leveraging decentralized learning, multi-agent coordination, and continuous telemetry-driven reasoning, surpassing centralized and static orchestration approaches	>25% compared to SotA	Centralized and static orchestration mechanisms	Obj.2
T2.5	Increase the robustness of the algorithms, ensuring consistent performance even under uncertain or noisy conditions.	Improve robustness over centralized or non-coordinated approaches by using decentralized AI and multi-agent strategies, ensuring consistent performance under uncertain and noisy conditions.	>25% compared to SotA	Centralized or non-coordinated approaches	Obj.2
T3.1	Number of trustworthy	Enable trustworthy identity and trust	>=3	-	Obj.3

	identity and trust management processes enabled by smart contracts.	management processes using smart contracts, protecting against unauthorized access and malicious behavior.			
T3.2	Accuracy of user and device verification and authentication.	EMPYREAN will improve the accuracy of user and device verification and authentication compared to traditional identity and access management, reducing risks of unauthorized access.	> 99%;	Traditional identity and access management mechanisms	Obj.3
T3.3	Reduction of privacy violation incidents in data sharing.	Reduce privacy violation incidents in data sharing compared to conventional centralized or ad-hoc mechanisms, protecting sensitive information from exposure.	> 50%;	Conventional centralized or ad-hoc mechanisms	Obj.3
T3.4	Time reduction to read/write data when storing data purely on the edge compared to storage on the cloud.	EMPYREAN will decrease read/write latency for data stored on the edge compared to standard cloud storage, mitigating delays associated with cloud dependency.	by 40%	Storage on the cloud	Obj.3
T3.5	Ability to access data stored on the edge when the link to the cloud is severed.	Enable access to data stored on the edge even if the cloud link is lost, unlike typical cloud-dependent storage, ensuring uninterrupted service and data availability.	-	Storage on the cloud	Obj.3
T4.1	Increase small-message transfer performance measured at the application level.	Increase small-message transfer performance at the application level compared to conventional CPU-managed data transfers	by 3x	CPU-managed data transfers	Obj.4
T4.2	Improve the RDMA programming efficiency of edge applications.	EMPYREAN will improve RDMA programming efficiency for edge applications compared to traditional	-	Typical RDMA usage paradigm	Obj.4

		RDMA usage			
T4.3	Decrease the wired overhead over today's protocols like MQTT and Kafka.	Decrease wired communication overhead compared to current protocols like MQTT and Kafka, optimizing network utilization for IoT and edge workloads.	by 50%	Protocols like MQTT and Kafka	Obj.4
T4.4	Ensure that the amount of erasure coded data retrieved for a query scales linearly.	EMPYREAN will ensure that the amount of erasure-coded data retrieved scales linearly with query size.	-	-	Obj.4
T4.5	Limit on the overhead incurred by erasure coded techniques	Provide an upper limit on the overhead incurred by erasure coded techniques, that is either constant or a linear function.	-	-	Obj.4
T5.1	Reduce the development time of continuum-native applications	Total engineering hours to deliver a continuum-native application from concept to deployment. Compares efficiency against the current SoTa tools and workflows.	>20% decrease compared to SotA	Traditional Cloud related tools such as AWS	Obj.5
T5.2	Number of supported hardware architectures for seamless deployment of an application.	Evaluates portability and adaptability across heterogeneous environments (e.g., CPU, GPU, FPGA, edge devices).	>3	Support for 3 hardware architectures (e.g., x86 CPU, NVIDIA GPU, ARM-based edge device)	Obj.5
T5.3	Reduce memory and space required for deploying application in resource-constrained IoT/Edge devices.	Evaluates optimization efficiency through lightweight frameworks, or deployment pipeline improvements.	>70% decrease of footprint	Traditional deployment tools such as Docker	Obj.5
T5.4	Offload acceleration functionality to nearby devices.	Evaluates improvements in distributed processing efficiency and workload balancing across the continuum.	>1 IoT device, >3 Edge devices		Obj.5

6 Requirements Analysis

This section presents the EMPYREAN requirements derived from a comprehensive analysis of the project Use Cases (UCs) and platform components. The following principles were applied to ensure optimal definition and clarity of the requirements:

- ***Traceability***: Each requirement must be traceable to an operational need. Once defined, it receives a unique identifier that allows the software design, code, and test procedures to be precisely traced back to the requirement.
- ***Clarity***: Requirements should be unambiguous. Vague and general statements must be avoided. Descriptions need to be clear, specific and singular.
- ***Measurability***: Requirements should be measurable, either quantitatively or qualitatively.
- ***Uniqueness and Consistency***: Requirements must be uniquely identified, consistent, and compatible with each other.
- ***Feasibility***: Requirements should be feasible and design-free, reflecting what the system needs to accomplish rather than how it should be designed.

EMPYREAN requirements are organized systematically to facilitate efficient processing and transformation into architectural decisions. All partners used a template to maintain consistency and uniformity across all requirements. The template includes the following fields:

- ***Requirement ID***: Each requirement is assigned a unique ID that encapsulates its type (i.e., Functional (F) or Non-Functional (NF)), key area it belongs to, and numbering sequence.
- ***Priority***: Requirements are prioritized to indicate their importance and order of implementation. Priority levels: Must-have: mandatory, Should-have: desirable, Could-have: optional, Will-not-have: possible future enhancement.
- ***Title***: A short but descriptive title for the requirement.
- ***Stakeholders and Actors***: Stakeholders or systems interacting with the requirements as well as the actors within the consortium that are involved in the requirement.
- ***Description***: Provides a detailed explanation of the requirement.
- ***Rationale/Goal***: Explains the reason for the requirement and its intended outcome.
- ***Relation to UCs***: Describes the EMPYREAN use cases whose operation is related to the specific functionality.
- ***Acceptance Measures***: Defines criteria to evaluate the successful implementation and performance of the requirement.
- ***Dependencies***: Lists other requirements or components that are related or necessary for the fulfilment of the requirement.

Based on the EMPYREAN goals and technical objectives, the requirements are grouped into 7 (seven) categories, encompassing both the project use cases requirements and the innovative technologies envisioned to enable a cognitive continuum, integrating intelligence and automation to achieve more efficient data processing.

These categories include general requirements, EMPYREAN Associations requirements, security and trust requirements, data and computing management requirements, decentralized intelligence requirements, service orchestration, distributed application development and deployment requirements, and integration and platform development requirements.

A total of 63 requirements have been defined, representing what the EMPYREAN platform components should address to support an Association-based continuum and the project use cases. This continuum aims to facilitate trustworthy, cognitive, and AI-driven associations of IoT devices and edge resources for efficient data processing. By systematically organizing and prioritizing these requirements, EMPYREAN ensures a clear and actionable roadmap for developing a robust, secure, and intelligent platform. Moreover, the requirements will constitute the basis of the initial architecture of the EMPYREAN project, detailed in deliverable D2.2 “Initial Release of EMPYREAN Architecture” (M7).

6.1 Functional Requirements

6.1.1 General Requirements

This section presents the general functional requirements for the EMPYREAN platform, which are indispensable parts of the designed solution. They describe the desired functionality the designed platform, and its integrated software mechanisms must offer to the end users. In addition, there are requirements to ensure the future exploitation of the EMPYREAN developments, their ability to support cognitive and edge intelligence beyond current data infrastructure, and enabling international collaboration with trusted partner regions to showcase the project advancements.

Table 4: Analysis of general requirements

Requirement ID	F_GR.1	**Priority**	Must-have
Requirement Title	Federate heterogeneous and distributed IoT, edge and cloud resources.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	All
Description	The EMPYREAN platform should seamlessly, autonomously, and efficiently integrate heterogeneous resources from various administrative (e.g., private, public, self-hosted) and technological domains. These resources include IoT devices, on-premise systems, deep and far edge resources, and multiple cloud platforms. The platform must intelligently leverage this diverse array of resources to execute dynamic and highly demanding cloud-native applications.		

Rationale/Goal	By unifying IoT, edge, and cloud computational and storage resources through an automated and self-managed approach, EMPYREAN will ensure that data and processing for extremely low-latency services remain close to their source. Meanwhile, computationally and data-intensive applications will be intelligently distributed across a diverse set of edge and cloud platforms.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	EMPYREAN orchestration and deployment mechanisms are aware of the available resources across the continuum.
	Cloud-native application deployment across IoT-edge-cloud continuum.
	EMPYREAN platform must respect data sovereignty and privacy requirements.
Dependencies	EMPYREAN decentralized intelligence and AI-enabled application development and deployment mechanisms.
	Availability of telemetry information.
	Abstraction models for cloud-native applications and infrastructure resources.

Requirement ID	F_GR.2	**Priority**	Must-have
Requirement Title	Enable collaborative autonomy in the IoT-edge-cloud continuum.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	All
Description	The EMPYREAN platform must deliver a collaborative autonomous computing ecosystem through the EMPYREAN Associations, leveraging heterogeneous resources, various providers, and diverse connectivity types across the IoT-edge-cloud continuum. An Association comprises IoT devices, multilayer edge resources, and federated clouds, forming a trusted, autonomous, and dynamic entity. Utilizing AI algorithms, each Association should elastically and flexibly allocate resources, manage workload, and handle data. Furthermore, the Associations must enable the seamless deployment of hyper-distributed applications within the continuum, supporting their autonomous and self-driven adaptations.		
Rationale/Goal	EMPYREAN envisions transforming the highly heterogeneous and distributed IoT-edge-cloud continuum into a cohesive, borderless infrastructure through multiple collaborative Associations. This Association-based continuum will provide applications and IoT devices with a unified, adaptable, efficient, reliable, and trustworthy virtual execution environment.		
Relation to UCs	All EMPYREAN use cases.		

Acceptance Measures	Efficient usage of computational, storage, and network resources across the continuum.
	Enhanced scalability, reliability, and responsiveness to application and infrastructure changes.
Dependencies	Autonomous distributed decision-making and collaborative intelligence.
	Decentralized security and privacy mechanisms for verifying identities and controlling access to resources.
	Well-defined and documented APIs for seamless interaction between IoT devices, edge nodes, and cloud services.

Requirement ID	F_GR.3	**Priority**	Must-have
Requirement Title	Encompass autonomous and continuous control loops.		
Stakeholders	Service Providers, End Users	**Actors**	All
Description	Several key operations of the EMPYREAN AI-enabled distributed control and management plane, which support the Association-based operations, should employ autonomous and semi-autonomous approaches for self-adaptation and self-healing. The platform must include multiple distributed control loops operating at the Association level. These loops will utilize data-driven mechanisms to sense (detect what is happening), discern (interpret senses), infer (understand implications), decide (choose actions), and act (execute actions), over an infinite time horizon.		
Rationale/Goal	These mechanisms will drive the automated and cognitive operation of Associations, as well as the orchestration of workloads and data management across and within the EMPYREAN Associations. The distributed control loops will underpin collaborative autonomy in the IoT-edge-cloud continuum by abstracting the underlying complexity and heterogeneity of resources, while co-optimizing resource utilization and simultaneously enhancing scalability and resiliency.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Automatically and intelligently balance workload across the Association-based decentralized computing environments.		
	Detect critical situations and reactively/proactively trigger re-optimization actions.		
	Support self-adaptation of Associations and deployed workloads to automate and optimize the Association-based continuum operation.		
Dependencies	Cognitive orchestration and service assurance mechanisms.		
	Availability of telemetry data and historical monitoring information.		

Requirement ID	F_GR.4	Priority	Must-have
Requirement Title	Provide seamless deployment of hyper-distributed cloud-native applications across a collaborative IoT-edge-cloud continuum.		
Stakeholders	Service Providers, End Users	Actors	All
Description	The platform should provide an abstraction layer that ensures seamless workload portability across the different platforms within the EMPYREAN-based continuum. EMPYREAN should facilitate resource interoperability and transparent deployment of applications' workload and data across the entire IoT-edge-cloud continuum.		
Rationale/Goal	The seamless deployment will enable EMPYREAN orchestration mechanisms to cater for application constraints, while calibrating the configuration of available resources. It will also guarantee a level of interoperability and portability, increasing the potential impact of the developed mechanisms. Moreover, developers will be able to focus solely on business logic for their applications.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Seamless deployment of workload across the EMPYREAN continuum by supporting variety of process architectures and environments (e.g., containers, microVMs, WebAssembly).		
	Provide abstractions for cloud-native application deployment over seamlessly integrated heterogeneous computing resources.		
	Support develop once, deploy everywhere paradigm for micro-services and serverless based cloud-native applications.		
Dependencies	EMPYREAN workflow-based application design and packaging mechanisms.		
	EMPYREAN low-level application deployment mechanisms.		

Requirement ID	F_GR.5	Priority	Must-have
Requirement Title	Support hyper-distributed, highly-demanding, and dynamic applications from diverse domains.		
Stakeholders	Application Developers, Service Providers, End Users	Actors	All
Description	The EMPYREAN platform must provide a novel ecosystem of cloud-based technologies to support future hyper-distributed applications. It must deliver intelligence on demand along with dynamic scalability (when/where needed) and automatic data interconnection.		

Rationale/Goal	Demonstrate the innovative capabilities of the EMPYREAN platform in supporting the three diverse, hyper-distributed, dynamic, and high-performance demanding project use cases. These UCs need more computational power and intelligence on the edge while operating in various environments (e.g., industry, warehouse, field) and including robots, drones, and other sensors.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Successful demonstration of improved operation for three UCs by leveraging EMPYREAN innovations.
	Provide seamless deployment, cognitive orchestration, service assurance, enhanced security, and intelligent management to the three UCs.
Dependencies	EMPYREAN SDK.
	EMPYREAN security, trust, and data and computing management mechanisms.
	EMPYREAN decentralized intelligence and AI-enabled application development and deployment.

Requirement ID	F_GR.6	**Priority**	Must-have
Requirement Title	Provide monitoring for cloud-native applications and heterogeneous infrastructure resources.		
Stakeholders	Application Developers, Service Providers, End Users	**Actors**	All
Description	The EMPYREAN platform must integrate advanced telemetry mechanisms to automatically discover and monitor the available heterogeneous resources automatically. In addition, it should dynamically and transparently monitor all deployed cloud-native applications and serverless workloads across the Association-based continuum.		
Rationale/Goal	Orchestration, autoscaling, and service assurance mechanisms must have access to detailed infrastructure resource information (e.g., type, quantity, capabilities) and comprehensive monitoring data to provide informed decision-making processes.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Collect detailed telemetry data from heterogeneous infrastructure resources across the IoT-edge-cloud continuum.		
	Collect performance metrics for the deployed cloud-native applications.		
	Provide historical telemetry data.		
Dependencies	Capabilities and exposed APIs of individual IoT, edge, and cloud platforms.		
	Availability of telemetry probes.		

Requirement ID	F_GR.7	**Priority**	Must-have
Requirement Title	Energy and power aware operation for optimal power management, energy efficiency and ecological sustainability.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	All
Description	EMPYREAN platform should provide power consumption - aware operation of applications considering the battery-based function of devices in the automation process. Furthermore, it should enable intelligent green application management to promote optimal energy use and ecological sustainability across the IoT-edge-cloud continuum. EMPYREAN decision-making mechanisms should jointly consider performance and energy efficiency for the workload placement and management of the Associations. They should also account for the energy sources powering the devices and the resources (e.g., battery or through renewable energy sources), their charge states and their overall “greenness”, prioritizing green resources.		

Rationale/Goal	EMPYREAN aims to contribute to Europe's edge-to-cloud decarbonization through energy-aware workload and data balancing to reduce environmental footprint by intelligently leveraging edge resources and integrating environmental considerations into the cognitive orchestration mechanisms.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Reduce energy consumption on Associations compared to standard execution.
	Orchestrate applications' workload and data based on energy-aware algorithms.
Dependencies	Energy-aware resource orchestration algorithms.
	Ability to measure and/or estimate the energy consumption for key infrastructure resources.
	Availability of data (e.g., capabilities, prices) for the available renewable and non-renewable energy sources.

6.1.2 Associations Requirements

This section focuses on the requirements for the EMPYREAN IoT-Edge Associations, or simply Associations, which are the fundamental building blocks of the envisioned collaborative IoT-edge-cloud continuum. These requirements are elicited to ensure the autonomous and seamless management and operation of the Associations by leveraging an AI-enabled management scheme. Additionally, they outline the core functionalities that need to be implemented by the EMPYREAN Aggregator. The Aggregator plays a crucial role in supporting the formation, coordination, and collaboration of Associations, utilizing the advanced technologies developed by EMPYREAN.

Table 5: Analysis of EMPYREAN IoT-Edge Associations requirements

Requirement ID	F_ASSOC.1	**Priority**	Must-have
Requirement Title	Combine heterogeneous computational and storage resources and different connectivity resources.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	All

Description	EMPYREAN Associations must operate over a federation of heterogeneous IoT, edge, and cloud resources and services that may belong to several users or organizations. These infrastructure resources are interconnected utilizing different connectivity means/types. Through an Association, resources with different capabilities and/or roles are shared between the participant users and seamlessly and cognitively combined to create a unified virtual execution environment. In this way, EMPYREAN Associations should transform the IoT-edge-cloud continuum into a set of autonomous, self-organized, and collaborative continuums, namely the Associations continuum. The Association-based continuum should be able to cognitively orchestrate resources and efficiently distribute and balance data and workload within an Association and/or across multiple Associations.
Rationale/Goal	A hyper-distributed cloud-native application will be able to seamlessly leverage infrastructure resources with varying capabilities and availability. The resources of an Association will be shared between the participant users. The collaboration of Associations will enable the intelligent and adaptive distribution of applications' workloads to optimize overall performance and resource utilization. The Associations will be also empowered by automated tools and mechanisms for seamless interconnection, efficient data processing of ML-workloads, and secure distributed data storage.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Increase statistical multiplexing in resource usage through Associations.
	Energy-aware utilization of infrastructure resources.
	Limit data transfer to centralized cloud platforms.
Dependencies	IoT and edge resources integration mechanisms.
	Coordination and management mechanisms for hybrid edge and cloud environments.
	Data exchange and storage across IoT devices and multiple edge and cloud layers.

Requirement ID	F_ASSOC.2	**Priority**	Must-have
Requirement Title	Facilitate secure onboarding of new IoT devices, robots and edge/cloud resources within the EMPYREAN control and management plane.		
Stakeholders	Infrastructure Owners	**Actors**	All

Description	The Associations must automate the addition and removal of multiple IoT devices, robots, and edge/cloud nodes, leveraging the EMPYREAN AI-enabled control and management plane for seamless integration. New nodes should be able to automatically communicate with existing nodes both within their own Association and across other Associations, fostering a cohesive and interoperable execution environment. Newly added nodes should become instantly available to the orchestration mechanisms for workload placement and data storage. IoT devices and robots will also become available as data sources that will feed ML mechanisms served by an Association. This immediate availability is crucial for maintaining optimal performance and resource utilization across the continuum. An EMPYREAN Association should expose the appropriate APIs for registering and removing resources and other nodes. By linking IoT devices, robots and edge/cloud nodes to a specific Association, the EMPYREAN platform will enhance their discoverability, making them securely identifiable and manageable within the continuum.
Rationale/Goal	By automatically addressing these aspects, EMPYREAN platform will facilitate the seamless and secure onboarding and offboarding of IoT devices, robots and edge/cloud resources within the Associations. Moreover, it will support the dynamic modification of the resources that compose a certain Association based on various operational criteria. Thereby, the implementation of this requirement will ensure a robust, trustworthy, scalable, and cognitive IoT-edge-cloud continuum. This will not only streamline operations but also facilitate the continuous evolution and scaling of the EMPYREAN platform to meet future and operational challenges.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Automate onboarding and offboarding processes, minimizing human intervention.
	Provide resource classification based on reliability, trustworthiness, availability, and connectivity.
	Facilitate the up-to-date IoT-edge-cloud continuum topology provision to the cognitive orchestration and service assurance mechanisms.
	Allow to specify the necessary configuration to ensure the appropriate onboarding of new nodes (e.g., IoT devices and robots) in the overall EMPYREAN platform.
Dependencies	Security, trust, and privacy preserving mechanisms.
	Dynamic connectivity between heterogeneous edge-cloud resources.
	Automated configuration management and monitoring

Requirement ID	F_ASSOC.3	**Priority**	Must-have
Requirement Title	Constitute a secure and trustworthy execution environment.		

Stakeholders	Infrastructure Owners, Service Providers, End Users	**Actors**	NUBIS
Description	EMPYREAN will implement a secure and trustworthy execution environment leveraging several key technologies and approaches. This will include designing and implementing a distributed trust management framework, trust propagation, and verification, combining trusted computing & storage, smart contracts, and distributed ledger technologies (DLTs). The platform will ensure secure boot and trusted execution, providing robust and flexible ecosystems capable of maintaining high levels of performance and security across various IoT-edge-cloud infrastructures. Additionally, EMPYREAN will use policy-based encryption techniques and identity management mechanisms to assure controlled access and data confidentiality.		
Rationale/Goal	EMPYREAN will provide a secure execution environment for dynamic, heterogeneous, and interconnected IoT devices and edge resources across the EMPYREAN platform, ensuring data and workload security, integrity, and confidentiality.		
Relation to UCs	All EMPYREAN use cases.		
	Implement a trust management and verification service using trusted resources, smart contracts, and DLTs.		
	Deploy and operate secure boot and trusted execution across multiple IoT-edge-cloud environments.		
	Use policy-based encryption and identity management to control access and maintain data confidentiality.		
	Achieve security performance metrics as defined in project's technical KPIs.		
Dependencies	Orchestration and management system.		
	Secure storage services.		
	Privacy-preserving mechanisms.		

Requirement ID	F_ASSOC.4	**Priority**	Must-have
Requirement Title	Support autonomous operation and enhance resiliency across the continuum.		
Stakeholders	Service Providers, Application Developers, End Users	**Actors**	All

Description	The Associations must implement self-configuration and self-optimization mechanisms to support the autonomous management of the distributed IoT, edge, and cloud resources across the continuum. Each Association should include its own data-driven optimization and configuration mechanisms to provide closer control over the specific part of the overall infrastructure it oversees. Moreover, an Association should adapt to abrupt resource availability and connectivity changes, with the respective orchestration mechanisms to automatically reallocate affected workloads within an Association and/or across Associations. In case, cloud resources are not available due to connectivity or other reasons an Association or the Associations-continuum should support the efficient processing and storage of data generated by IoT devices and robots.
Rationale/Goal	Associations should be able to operate independently and maintain functionality under various conditions (e.g., lost connection with central cloud platforms, limited connectivity within IoT devices and deep edge nodes). This requirement is essential for ensuring that systems can continue functioning effectively despite disruptions, varying workloads, or changing environments. Additionally, Associations should include multi-agent and multi-objective optimization mechanisms to continuously optimize performance metrics such as latency, energy consumption, and resource utilization through AI-driven analysis and decision-making.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Support autonomous adaptations to continuum dynamicity.
	Increase reliability in the edge and support for self-healing functionality.
	Implement AI-enabled management schemes based on adaptive learning and predictive analytics.
	Provide fault tolerance and recovery.
Dependencies	Dynamic application deployment, support for application-level adaptations.
	Adaptive resource management and self-configuration capabilities.
	Advanced analytics and AI, decentralized decision-making.

Requirement ID	F_ASSOC.5	**Priority**	Should-have
Requirement Title	Provide low and predictable latency for hyper-distributed applications.		
Stakeholders	Service Providers, Application Developers, End Users	**Actors**	All

Description	Providing low and predictable latency for hyper-distributed applications is essential for optimizing performance, ensuring operational efficiency, and gaining a competitive advantage. By integrating edge computing, employing intelligent workload placement, implementing decentralized and message-centric data communication, providing efficient distributed edge storage, monitoring latency, and software-defined edge interconnection for distributed computations, this requirement can be effectively achieved. These strategies collectively contribute to a robust infrastructure capable of supporting the demanding requirements of modern hyper-distributed applications.
Rationale/Goal	The integration of the above strategies with automated service assurance mechanisms will empower the EMPYREAN platform to self-adjust, providing a robust framework for efficient orchestration and deployment of latency-sensitive and safety-critical applications in the complex and dynamic environment of the IoT-edge-cloud continuum.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Facilitate network- and latency-aware scheduling policies.
	Leverage data locality by distributing processing closer to where data is generated.
	Reduce the amount of data sent across the network by processing as much as possible locally and exploiting caching techniques.
Dependencies	Availability of historical network telemetry data.
	Configuration capabilities and monitoring data from the underline network infrastructure.
	Optimized network protocols for low latency and reduced payload size.

Requirement ID	F_ASSOC.6	**Priority**	Must-have
Requirement Title	Provide inter-Association communication and exchange of events.		
Stakeholders	Service Providers, Application Developers	**Actors**	All
Description	The EMPYREAN should enable the communication of multiple EMPYREAN Associations to exchange information, data, policies, workloads, events, and local views about the available resources and deployed applications. An Association should act as a gateway for information, data, and events to resources that form the Association and to inter-Association communication. This requirement is critical to the overall collaborative operation of the distributed and cognitive orchestration and service assurance mechanisms of the EMPYREAN platform.		

Rationale/Goal	The platform's collaborative orchestration, deployment, and control management functionalities are distributed across the IoT-edge-cloud continuum, ensuring both cooperative operation and scalability. To achieve this, mechanisms within each Association must effectively exchange information, data, and workloads with their counterparts in other Associations. This interconnectivity is vital for maintaining the robust and efficient operation of the EMPYREAN platform.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Ability to deliver events to multiple endpoints.
	Seamless exchange of data and requests among the EMPYREAN Associations.
	Response time should be acceptable from the other services.
Dependencies	Availability of telemetry mechanisms for information collection.
	Service assurance mechanisms for information correlation, analysis, and event identification.
	Support for asynchronous communication and data transfer across the EMPYREAN platform.

Requirement ID	F_ASSOC.7	**Priority**	Must-have
Requirement Title	Data-driven seamless workload and data migration across the Associations.		
Stakeholders	Service Providers, Application Developers, End Users	**Actors**	All
Description	The Associations must provide a collaborative environment for deploying hyper-distributed applications and ML-based workloads efficiently over the continuum, along with the necessary data placement. Moreover, an Association must support the autonomous and data-driven workload and data migration across its resources as well as between other Associations.		
Rationale/Goal	An Association should provide the appropriate mechanisms to automatically migrate part of the overall application workload within its resources and/or across multiple Associations. Additionally, they should automatically and transparently (for the end users) migrate data within the distributed infrastructure to ensure that the required data are always available to workloads. These mechanisms will facilitate the EMPYREAN distributed and AI-enabled control and management plane to maintain the actual operation state close to the desired state.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Reduce reliance on cloud and increase edge utilization.		
	Provide intelligent placement and workload balancing optimization.		
	Workload resumes its operation without user intervention.		
Dependencies	Resource orchestration and data placement algorithms.		

	Interoperable container runtimes.
	Cloud-native application deployment mechanisms over heterogeneous edge/cloud platforms.

Requirement ID	F_ASSOC.8	**Priority**	Must-have
Requirement Title	Aggregators must maintain a catalogue of the Association resources.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers, End Users	**Actors**	All
Description	The Aggregator should implement a digital catalogue to persist information about an Association's available resources, services and data. This catalogue should provide a registry to publish available resources, platform services and data in order to facilitate the management of Association resources, the deployment of application workloads, and other inter-Association operations. Furthermore, it should provide resource classification, enabling the EMPYREAN platform to catalogue the characteristics, reliability, and trustworthiness of system resources and services in order to take them into consideration during the orchestration operations.		
Rationale/Goal	These functionalities are critical for realizing the Association-based continuum as they will provide topology awareness within the EMPYREAN platform. Moreover, they will facilitate the onboarding of new infrastructure resources and the deployment of new hyper-distributed services by enabling their registration and discovery via the appropriate APIs.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Facilitate the Associations' management and collaboration across the IoT-edge-cloud continuum.		
	Implement API to manage catalogue entries.		
	Support filtering stored information based on various criteria such as specific characteristics, latency, ownership, energy consumption, and cost.		
Dependencies	Availability of persistent data model for provider resources and services.		
	Service discovery and monitoring capabilities of IoT devices, edge and cloud resources and services.		

Requirement ID	F_ASSOC.9	**Priority**	Must-have
Requirement Title	Aggregators must dynamically discover resources within the registered infrastructures and detect events.		
Stakeholders	Infrastructure Owners, Service Providers, End Users	**Actors**	All

Description	The EMPYREAN platform should include methods to automatically and dynamically discover the registered infrastructures. The Aggregator must provide decentralized autonomous discovery mechanisms for infrastructure description meta data such as the available IoT/edge/cloud devices, meta-data information, and their capabilities.
Rationale/Goal	The orchestration, deployment, and service assurance mechanisms will always retrieve an up-to-date list of available infrastructure resources to effectively perform their operations. Furthermore, the service assurance mechanisms will leverage the provided information to initiate necessary re-optimization procedures in response to any changes in resource availability.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Provision of detailed inventory parameters (e.g., amount, type, capabilities, operation state) of available computational and storage resources within an Association.
	Dynamic detection of changes in resources that constitute an Association over time.
Dependencies	Capabilities and exposed APIs for the individual IoT devices, edge, and cloud platforms.
	Adoption of a common resource description schema.

Requirement ID	F_ASSOC.10	**Priority**	Must-have
Requirement Title	Aggregators must maintain the state of the Association.		
Stakeholders	Infrastructure Owners, Service Providers, End Users	**Actors**	All
Description	Each Aggregator must maintain the detailed state of the resources and deployed workload for the Association it manages. For example, IoT devices and edge resources are part of the Association. This information is critical for the efficient management of the Associations, the cognitive and speculative orchestration of resources, as well as the AI-driven adaptability within the EMPYREAN platform.		
Rationale/Goal	This requirement ensures that orchestration, deployment, and service assurance mechanisms always have the most current information regarding the Association's capacity, performance, and status. By consistently updating their knowledge base of available resources, the above mechanisms can proactively adjust to varying conditions, thus ensuring continuous and reliable service delivery. This dynamic adaptability not only helps in maintaining service levels but also enhances the efficiency and resilience of the overall Association-based continuum.		

Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Collect performance metrics for deployed application and utilization for infrastructure resources.
	Response time and monitoring granularity should be acceptable for the appropriate operation of the orchestration and service assurance mechanisms.
	Ability to provide historical monitoring data to EMPYREAN AI-enabled mechanisms.
Dependencies	Capabilities and exposed APIs for the individual IoT devices, edge, and cloud platforms.
	Long-term storage of collected monitoring data.

6.1.3 Security and Trust Requirements

This section outlines the initial functional requirements for EMPYREAN's security and trust mechanisms. These requirements aim to ensure that EMPYREAN Associations operate as secure and trusted execution environments. Trust between data generating and data processing entities will be continuously validated using distributed trust services, identity and access management mechanisms, and trusted boot and execution technologies.

Table 6: Analysis of security and trust requirements

Requirement ID	F_ST.1	**Priority**	Must-have
Requirement Title	Decentralized Identity Management		
Stakeholders	Infrastructure Owners, Service Providers, End Users	**Actors**	UMU, All
Description	EMPYREAN will implement decentralized identity management mechanisms utilizing Decentralized Identifiers (DIDs) and Distributed Ledger Technologies (DLTs). This will ensure self-sovereign identity for users and devices, giving them more control over their data and reducing the risk of unauthorized access.		
Rationale/Goal	Decentralized identity management is a key component of the EMPYREAN vision for a collaborative and autonomous computing ecosystem. By using DIDs and DLTs, the project aims to create a more secure, private, and user-centric approach to identity management, aligning with the principles of self-sovereignty and data protection.		
Relation to UCs	The implementation of decentralized identity management will enhance security and privacy in all use cases within the EMPYREAN platform. It will be particularly beneficial in scenarios requiring secure authentication and		

	authorization, ensuring that only authorized entities can access sensitive resources and data.
Acceptance Measures	Successful implementation of DIDs and their integration with DLTs for secure and transparent identity management. Demonstration of self-sovereign identity capabilities for users and devices within the EMPYREAN ecosystem.
	Achieving a user satisfaction rate of over 90% regarding the ease and security of the identity management process.
Dependencies	Integration with Privacy and Security Manager (UMU) for identity management functions.
	Compatibility with existing Distributed Ledger Technologies (DLTs). Utilization of the P-ABC library (UMU) for cryptographic operations related to attribute-based credentials.

Requirement ID	F_ST.2	**Priority**	Must-have
Requirement Title	Privacy-Preserving Authentication and Authorization		
Stakeholders	Infrastructure Owners, Service Providers, End Users	**Actors**	UMU, All
Description	EMPYREAN will implement a privacy-preserving authentication and authorization system based on attribute-based verifiable credentials (VCs). It will allow users and devices to authenticate themselves and authorize access to resources based on their attributes, without revealing unnecessary personal information. The system will utilize cryptographic techniques such as Zero-Knowledge Proofs (ZKPs) to ensure that only the necessary attributes are disclosed during authentication and authorization processes.		
Rationale/Goal	The goal of this requirement is to enhance privacy and security in the authentication and authorization processes within the EMPYREAN ecosystem. By using attribute-based VCs and ZKPs, the project aims to minimize the disclosure of personal information while still allowing for secure and verifiable access control. This is crucial for building trust and ensuring data protection in the decentralized environment of EMPYREAN.		
Relation to UCs	This requirement is essential for use cases involving sensitive data and resources, where privacy-preserving authentication and authorization are critical. It will support scenarios where users and devices need to authenticate and authorize actions without compromising their privacy, ensuring secure access control in various applications within the EMPYREAN platform.		
Acceptance Measures	Successful implementation of attribute-based VCs and ZKPs for authentication and authorization. Demonstration of privacy-preserving		

	authentication and authorization processes within the EMPYREAN ecosystem. Verification that only necessary attributes are disclosed during these processes.
Dependencies	Integration with the Privacy and Security Manager (UMU) for managing verifiable credentials and zero-knowledge proofs.
	Utilization of the P-ABC library (UMU) for cryptographic operations related to attribute-based credentials.

Requirement ID	F_ST.3	Priority	Must-have
Requirement Title	Policy-Based Encryption		
Stakeholders	Infrastructure Owners, Service Providers, End Users	Actors	UMU
Description	EMPYREAN must implement policy-based encryption techniques, such as Ciphertext-Policy Attribute-Based Encryption (CP-ABE), to ensure controlled access and confidentiality of data. This involves delegating decryption rights to specific users, devices, or resources based on their attributes, enabling fine-grained access control.		
Rationale/Goal	Policy-based encryption is essential for protecting sensitive data and ensuring that it is only accessible to authorized entities based on their attributes. CP-ABE provides a flexible and scalable way to manage access control policies in the EMPYREAN ecosystem.		
Relation to UCs	UC1: Ensures encrypted data exchange between robotic cells and control systems, protecting sensitive operational data. UC2: Protects environmental data collected from sensors by encrypting it based on specific attributes. UC3: Encrypts logistics data to ensure it is only accessible by authorized entities.		
Acceptance Measures	Successful implementation of CP-ABE and other policy-based encryption techniques.		
	Demonstration of fine-grained access control based on attributes.		
	Integration of these mechanisms with the distributed trust management framework.		
Dependencies	CP-ABE Library: Used for implementing policy-based encryption.		
	Distributed Trust Management Framework.		

Requirement ID	F_ST.4	Priority	Must-have
Requirement Title	Automated Cyber Threat Analysis		
Stakeholders	Infrastructure Owners, Service Providers, Security experts	Actors	NEC

Description	This requirement entails developing advanced mechanisms for automated analysis of Cyber Threat Intelligence (CTI) sourced from various data channels. The project will encompass the entire lifecycle of CTI management, including the collection, storage, and curation of intelligence data from the Cyber Threat Alliance (CTA). A critical component of this initiative will be the creation and implementation of sophisticated algorithms designed to identify and track current global trends in exploited vulnerabilities and malware usage. These algorithms will enhance the ability to predict and respond to emerging cyber threats. The ultimate goal is to create a robust framework that supports continuous monitoring and analysis, thereby strengthening the overall defence against cyber threats.
Rationale/Goal	The objective is to enhance the platform's security by enhancing the security experts' knowledge of cyber threats in real-time. This requirement is crucial for maintaining the platform's integrity and protecting it against potential attacks. Robust threat detection and response mechanisms ensure continuous security and minimize the risk of breaches.
Relation to UCs	The GUI will be used to show security risks to the security experts.
Acceptance Measures	Integration with the CTA and MISP[147] platform.
	Presentation of an easy to use user GUI that can be used by security experts.
Dependencies	CTA platform.
	MISP platform.

Requirement ID	F_ST.5	Priority	Must-Have
Requirement Title	ML for Anomaly Detection and Cybersecurity		
Stakeholders	Infrastructure Owners, Service Providers, End Users	Actors	ICCS, NEC, RYAX, UMU
Description	EMPYREAN must incorporate ML mechanisms to detect risks or attacks on resources. This includes the development of a privacy-preserving Federated Learning (PPFL) system for anomaly detection and the analysis of multiple-source information to automatically learn the cybersecurity context and defend the Associations-based continuum.		
Rationale/Goal	Anomaly detection and cybersecurity measures are essential to protect the EMPYREAN ecosystem from threats and ensure its resilience. By leveraging ML and PPFL, EMPYREAN can proactively identify and respond to potential security risks.		
Relation to UCs	Directly related to UC1 and UC4, as both involve the use of machine learning for anomaly detection and cybersecurity measures, ensuring the		

[147] https://www.misp-project.org

	security and resilience of the EMPYREAN ecosystem. Additionally, UC3 benefits from ML mechanisms to ensure secure and reliable operations of autonomous robotic systems.
Acceptance Measures	Successful implementation of ML-based anomaly detection mechanisms.
	Development of a PPFL system for federated anomaly detection.
	Demonstration of the ability to learn the cybersecurity context and defend the continuum.
Dependencies	Monitoring services and telemetry data.
	Automation and resource management mechanisms.

Requirement ID	F_ST.6	**Priority**	Must-have
Requirement Title	Secure and Trusted Execution		
Stakeholders	Infrastructure Owners, Application Developers	**Actors**	NUBIS
Description	Establish a secure and trusted execution environment for data collection and processing across robotic machining cells (UC1), agricultural fields (UC2), and logistics operations (UC3). This involves using secure boot mechanisms, encryption, and other security measures to protect data and ensure the integrity of applications. For instance, ensuring secure execution environments in robotic machining cells can prevent unauthorized access and tampering with critical manufacturing data, while in agriculture, it can protect sensitive environmental data from being compromised.		
Rationale/Goal	To ensure the secure and reliable operation of applications, protecting sensitive data. Establishing a trusted execution environment is essential for maintaining data integrity and system reliability.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Boots and onboards correctly when firmware/application blob are signed.		
	Not continue the execution if the application running on the IoT device is not attested.		

6.1.4 Seamless Data and Computing Management Requirements

The efficient handling of ML workloads and hyper-distributed applications relies heavily on the interconnection and communication of the IoT devices with the computing and storage entities. Below, we present the initial functional requirements for seamless data and computing management within the EMPYREAN platform.

Table 7: Analysis of seamless data and computing management requirements

Requirement ID	F_DCM.1	**Priority**	Must-have
Requirement Title	Provide an S3-compatible storage service that encompasses the edge-cloud continuum.		
Stakeholders	Application Developers, End Users	**Actors**	CC, ICCS
Description	Applications running inside associations will need access to a simple to use storage service that abstracts away the complexities of managing storage resources across the edge-cloud continuum. Edge and cloud storage resources are typically highly heterogeneous and present a wide range of characteristics. System operators may create manual storage policies that account for this and try to balance user requirements such as availability, reliability, performance, and cost efficiency. However, given the complexity of matching individual user requirements to the available storage resources, an automated data distribution and storage policy creation mechanism is preferred. The service should also be prepared to run autonomously whenever the link to the cloud is severed. This requirement can only be met if sufficient storage resources exist in the association and an appropriate storage policy is utilized.		
Rationale/Goal	By hiding away and automating the complexities of managing the storage resources, application owners will benefit from a storage service that closely matches their requirements. The use of an S3-compatible API will make adoption significantly easier thanks to S3s widespread adoption and the wide range of tools available for it. By maintaining some level of autonomy, applications running at the edge, in an association, will still be able to utilize the storage service in case of network outages.		
Relation to UCs	This requirement captures the general need of the project's use cases to store data at the edge and in the cloud. It provides additional technical details and describes some of the requirements towards the underlying infrastructure.		
Acceptance Measures	Good coverage and compliance with Amazon S3's API.		
	Support for a wide range of cloud locations and edge storage resources.		
	Performance improvements when using edge compared to cloud storage locations.		
	Option to perform encryption and store associated keys solely on the edge, on privately managed infrastructure.		
	Maintain some level of functionality when the link to the cloud is severed.		
Dependencies	Telemetry service needs to capture state of storage resources.		
	Automated storage policy creation algorithms implemented by the orchestration service.		

Requirement ID	F_DCM.2	**Priority**	Must-have
Requirement Title	Provide an analytics-friendly erasure-coded IoT storage platform		
Stakeholders	Application Developers, End Users	**Actors**	CC, UC providers
Description	Storing large volumes of time series IoT data across the edge-cloud continuum presents several challenges. However, each has well-known solutions. Firstly, distributing the data to multiple cloud and/or edge locations is key to achieving good availability and reliability. Secondly, erasure coding offers much lower storage costs, compared to replication. Thirdly, to be able to support analytics workloads, data queries must be efficient in the amount of data that is retrieved from storage locations. This is especially important for cloud locations, which carry a significant data egress cost. Fourthly, large volumes of correlated data are good candidates for compression, widely used to decrease storage costs. The challenge is to offer all of these, sometimes incompatible, features as part of a single service.		
Rationale/Goal	All of the objectives highlighted above are desirable for a storage system designed for IoT time series data. Achieving them is technically difficult, especially in the case of a large volume of data with many dimensions, where the queries typically affect a small portion of the dimensions and stored records. This is a common scenario, especially when data is accumulated over a long period of time and must be analyzed to detect anomalies, short-time trends and so on. It is also applicable to cases when an ML model needs to be trained from a specific subset of a large dataset (e.g. after running PCA to determine the most important dimensions).		
Relation to UCs	The requirement captures the need of the use cases related to storing and later running analytics workloads on time-series data.		
Acceptance Measures	Store time-series data in a distributed fashion across edge and cloud locations.		
	Evaluate the possibility to use compression.		
	Low overhead of data that needs to be transferred to evaluate a query, compared to a traditional replicated approach		
	Good scaling (ideally linear) of the data transferred w.r.t. the retrieved dimensions for queries.		
Dependencies	One or more use cases that require a large volume of multi-dimensional time series data to be stored and queried.		

6.1.5 Decentralized Intelligence Requirements

To achieve EMPYREAN's ambition for an Association-based, cognitive and AI-driven continuum, an intelligent orchestration and management layer is essential. This layer must operate continuously and autonomously, optimizing and refining key application and system metrics such as quality of service, efficiency, scalability, and resiliency. Bellow, we outline the initial functional requirements for EMPYREAN's decentralized intelligence mechanisms. These requirements ensure that computational tasks and data are dynamically and efficiently assigned based on current operational needs while the Associations operate as intended.

Table 8: Analysis of decentralized intelligence requirements

Requirement ID	F_DI.1	**Priority**	Must-have
Requirement Title	Decentralized decision-making, speculative and multi-objective resource orchestration.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	ICCS, RYAX
Description	EMPYREAN should develop decentralized decision-making mechanisms to enable (a) the placement of applications' workload in the Associations and (b) the load balancing of processing and data among Associations. Speculative resource orchestration algorithms are required to balance between centralized and decentralized solutions for the efficient and autonomous operations of the Association-based continuum.		
Rationale/Goal	EMPYREAN's vision for an autonomous and collaborative operation within the IoT-edge-cloud continuum requires more local decisions and a collective logic to lead to system-wide welfare optimality.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Novel multi-agent and multi-objective resource allocation and service orchestration optimization algorithms.		
	Improve overall performance, enhance scalability and reliability.		
	Reduce bottlenecks.		
Dependencies	Multi-criteria deployment of hyper-distributed applications.		
	Intelligent and autonomous decision-making agents.		
	Dynamic discovery and orchestration of edge-cloud resources.		

Requirement ID	F_DI.2	**Priority**	Must-have
Requirement Title	Multi-agent speculative intelligent resource orchestration across EMPYREAN Associations.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	ICCS

Description	EMPYREAN must follow a distributed approach for resource orchestration between Associations. To better address the multiple and conflicting objectives among Associations, EMPYREAN should also support multiple levels of co-operation (partial/no co-operation). The orchestration layer should include multi-agent algorithmic solutions that address the inherent uncertainty by exploiting the speculative dimension while considering critical constraints such as the estimations in spatial (processing requirements) and temporal (estimated processing time) dimensions and the heterogeneity and usage of resources.
Rationale/Goal	The orchestration layer must account for the diverse characteristics and constraints of each computing environment along with the emerging hyper-distributed applications that embrace dynamic and multi-phase workflows. Thus, the placement of the tasks must be considered jointly, while respecting their dependencies.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Ensure scalability for designed algorithms.
	Improve data-driven decision-making accuracy.
	Support multiple autonomous large-scale IoT-edge-cloud platforms.
Dependencies	Coordination, collaboration, and conflict resolution among the agents.
	Monitoring service and feedback loops to adjust and optimize performance continually.

Requirement ID	F_DI.3	**Priority**	Must-have
Requirement Title	Hierarchical orchestration and multi-objective optimization for cognitive resource orchestration within Associations.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	ICCS, RYAX
Description	Within an Association, EMPYREAN should implement a hierarchical orchestration approach, where the Association layer makes high-level allocation decisions, and the local orchestrator at each platform provides the final decisions. This will be achieved through multi-agent and multi-objective optimization, leveraging game theory and auction-based mechanisms enhanced by swarm intelligence. These techniques will ensure that workload requirements are met while effectively abstracting the underlying complexity.		
Rationale/Goal	By employing these approaches, EMPYREAN aims to create highly effective resource allocation mechanisms, enabling timely operations in complex, multi-technology, and multi-domain infrastructures.		
Relation to UCs	All EMPYREAN use cases.		
	Ensure real-time processing at the edge with minimal latency.		

Acceptance Measures	Support scalability of hyper-distributed applications
	Multi-criteria deployment of hyper-distributed applications.
Dependencies	Automation and resource management mechanisms.
	Monitoring services and telemetry data.

Requirement ID	F_DI.4	**Priority**	Must-have
Requirement Title	AI-enhanced data orchestration and storage resource management within and across Associations.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	ICCS, CC, ZSCALE
Description	EMPYREAN integrates diverse IoT, edge, and cloud storage resources to create a secure, distributed storage infrastructure within the platform. Deployed workloads bring their requisite data and frequently generate additional data that needs to be stored within the system. EMPYREAN's decentralized intelligent mechanisms should automate the data management by load-balancing data fragments at multiple locations, based on resource availability and the locality of the created or consumed IoT data.		
Rationale/Goal	EMPYREAN cognitive and distributed decision-making mechanisms should facilitate the efficient and transparent allocation and movement of data segments over the distributed edge and cloud storage resources when workloads are deployed or adapted due to re-optimizations.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Provide optimized data placement over multiple edge and cloud resources.		
	Select storage locations (edge, cloud) based on user preferences and current status of platform resources.		
	Implement dynamic allocation strategies to balance load and prevent overutilization or underutilization of storage resources.		
	Exploit data locality to prioritize edge nodes where required data is already stored.		
Dependencies	APIs for integration with the secure storage services across the continuum.		
	Decentralized and autonomous interconnection and data distribution.		
	Access to telemetry data for storage utilization and performance metrics such as latency, throughput, cost, etc.		

Requirement ID	F_DI.5	**Priority**	Must-have
Requirement Title	Energy-aware workload and data distribution mechanisms.		

Stakeholders	Infrastructure Owners, Service Providers	**Actors**	ICCS
Description	Energy-aware algorithms should be developed based on sophisticated AI/ML methods that consider multiple parameters (e.g., energy prices and storage, on-site availability of renewable energy sources) to enable predictable resource configuration, support workload placement in an energy-efficient manner, and automate load distribution and migration within the association-based continuum.		
Rationale/Goal	Energy-aware decision-making will dynamically assess the energy profiles of different nodes and strategically distribute workloads and data to achieve energy efficiency without compromising performance. It will enable more battery-powered IoT devices and energy-constrained edge nodes to join Associations and collaborate while integrating environmental considerations into the operation of the Association-based continuum.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Energy-specific algorithms for energy-efficient orchestration and load distribution.		
	Reduce energy consumption on Associations compared to standard execution.		
	Promote computing and storage resources that use green energy.		
Dependencies	Availability of energy consumption and resource utilization data.		
	Multi-criteria deployment of hyper-distributed applications.		

Requirement ID	F_DI.6	**Priority**	Must-have
Requirement Title	Monitoring and managing power and energy consumption in IoT devices and edge nodes.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	ICCS, RYAX, UMU
Description	Decentralized intelligence mechanisms should have access to comprehensive energy profiles and real-time energy consumption measurements for resources across the IoT-edge-cloud continuum. A distributed system should be implemented to collect the necessary metrics from various devices and nodes. These metrics should be stored in time-stamped solutions, ensuring data integrity and enabling historical analysis.		
Rationale/Goal	Operating multiple sensors and processing units, especially in remote areas, requires a reliable and sustainable power source. Thus, monitoring and managing the power and energy consumption of these devices is crucial to ensure their continuous and efficient operation. Battery life, energy efficiency, and the ability to harvest energy from renewable sources are critical factors to consider.		
Relation to UCs	All EMPYREAN use cases.		

Acceptance Measures	Access to energy consumption measurements for IoT and edge resources.
	Short- and long-term preservation of energy-related collected telemetry data.
Dependencies	Availability of energy consumption and resource utilization data.
	Ability to monitor service level objectives of deployed applications.

Requirement ID	F_DI.7	**Priority**	Must-have
Requirement Title	Decentralized and AI-enabled service assurance mechanisms.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	ICCS, NEC, RYAX, UMU
Description	EMPYREAN should provide autonomous adaptations for deployed applications and Associations through AI-enabled distributed service assurance mechanisms. These mechanisms should (i) adapt the resources within the Associations, (ii) provide dynamic load balancing of processing workload and data within and across Associations, and (iii) perform federated anomaly detection based on privacy-preserving federated learning.		
Rationale/Goal	Depending on the available telemetry information, these mechanisms will operate at different timescales. They will autonomously and continuously drive the orchestration mechanisms in re-optimizations and adaptations inside an Association using swarm intelligence and between federated Associations in a decentralized and multi-agent manner.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	React fast to rapid changes in computational and data demands to maximize the number of served demands.		
	Increase robustness, ensuring consistent performance even under uncertain or noisy conditions.		
	Protect the system from malicious activities that could exploit anomalies.		
Dependencies	Available monitoring data from the EMPYREAN telemetry mechanisms.		
	Dynamic application deployment and application-level adaptation.		

Requirement ID	F_DI.8	**Priority**	Must-have
Requirement Title	AI-enhanced self-healing for enhanced resiliency, adaptability, and autonomous operation.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	ICCS, RYAX, NEC

Description	EMPYREAN should address unexpected behaviours or failures within the Association-based continuum. Data-driven continuous analysis mechanisms based on swarm intelligence and machine reasoning should: (i) detect and categorize abnormal situations in applications/resources, (ii) mitigate resource fragmentation and connectivity issues, (iii) ensure continuous and reliable operation, and (iv) provide dynamic load balancing of processing workload and data within and across Associations.
Rationale/Goal	By implementing these data-driven mechanisms, the EMPYREAN platform can achieve robust anomaly mitigation, adaptability, and self-driven recovery, ensuring resilient and efficient operations in the face of unforeseen issues across the infrastructure. For example, if a node within an Association fails or leaves during the execution of an application, the platform must, if possible, maintain performance and minimize downtime with an automated recovery process.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	React fast to rapid changes in computational and data demands to maximize the number of served demands.
	Maintain optimal performance by quickly identifying and resolving anomalies.
	Learn from past anomalies and recovery actions to improve future responses.
Dependencies	Autonomous decision-making and predictive maintenance models.
	Monitoring service and telemetry data.

Requirement ID	F_DI.9	**Priority**	Must-have
Requirement Title	Autonomous and adaptive workload autoscaling.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	RYAX, ICCS
Description	EMPYREAN should be able to enable the dynamic and intelligent adjustment of computational resource boundaries to maximize task execution efficiency. Leveraging AI-enabled decision making, this requirement should ensure that resource allocation is continuously optimized responding to fluctuating workloads and varying performance demands. The system has to autonomously adjust the computational resources limits (i.e CPU, RAM) in such a way to cover the real needs of the executing tasks while fitting as many tasks as possible to the available resources.		
Rationale/Goal	The main idea for the adaptive workload autoscaling is to enable optimal bin-packing of tasks upon the resources by rightly setting the limits of the needed computational resources. This will eventually lead to the allocation and provisioning of less computing nodes to fit the executing workload. Based on that we can decrease the amount of used computing resources which will result in optimizing costs and energy consumption.		

Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Rightly adjusting the limits of tasks for efficient executions avoiding Out-Of-Memory errors and CPU throttling.
	Provide improvements in bin-packing of tasks minimizing the amount of needed resources for the execution of a workload.
Dependencies	Collecting resource usage (CPU, RAM) from executing tasks.
	AI-enabled mechanisms and autonomous decision-making.

6.1.6 Service Orchestration Requirements

The IoT-edge-cloud continuum is a complex and dynamic environment requiring robust, scalable, adaptable, and efficient service orchestration mechanisms. EMPYREAN will provide novel mechanisms for end-to-end cognitive orchestration and self-driven adaptability, streamlining the process of resource management in heterogeneous and federated IoT-edge-cloud infrastructures. Next, we present the initial functional requirements for the service orchestration mechanisms that aim to provide the envisioned cognitive, distributed, and autonomous orchestration functionality of the EMPYREAN platform.

Table 9: Analysis of service orchestration requirements

Requirement ID	F_SO.1	**Priority**	Must-have
Requirement Title	Continuum-native workflow-based application design considering dataflow programming and low-code techniques.		
Stakeholders	Application Developers, End Users	**Actors**	RYAX, ZSCALE
Description	EMPYREAN should be able to provide the tools to enable developers to build their applications destined to be executed on the continuum using intuitive, simple to use techniques. Continuum-native applications are expected to bring more complexities than Cloud applications mainly because of the usage of different infrastructures (IoT-Edge-Cloud) and their highly distributed nature. The specific requirement demands a certain abstraction and ease of use for the design of continuum-native applications. This should be enabled by proposing an application design based upon workflows and low-code techniques, which will allow a modular development of applications following a microservice/serverless approach to be adapted to EMPYREAN associations and aggregators concepts. Furthermore, the internal representation and communication of components in a workflow should be based upon a dataflow programming framework to better capture the distributed nature of continuum-native applications. Best practices and optimal configuration of applications should be considered. EMPYREAN should provide both Web and Command-line Interfaces for the application design techniques.		
Rationale/Goal	The main idea is to empower application developers with the right tools to design optimal continuum-native applications while reducing the time-to-		

	market. The tools need to be adapted to function according to the EMPYREAN associations and aggregators concepts. The techniques aim to minimize the needs for expert users to develop boilerplate code and for non-distributed systems experts to design continuum-native applications which can be executed efficiently.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Efficient design and deployment of Continuum-native applications upon EMPYREAN associations
	Minimize time-to-market for the development of optimal continuum-native applications.
	Enable workflow-based application design enhanced with lower-level dataflow programming.

Requirement ID	F_SO.2	**Priority**	Must-have
Requirement Title	Deployment objectives (SLOs) definition for continuum-native applications		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers, End Users	**Actors**	RYAX, ICCS
Description	EMPYREAN should be able to establish precise and measurable performance metrics that guide the deployment and operation of continuum-native applications across the continuum. This involves setting specific SLOs related to latency, throughput, availability, reliability, and energy efficiency for each application workflow, ensuring that every segment of the deployment upon the continuum meets the designated performance standards.		
Rationale/Goal	Given the diverse and distributed nature of resources and applications within the continuum, establishing clear and measurable SLOs is crucial for aligning resource management strategies with application performance needs. This requirement aims to provide a robust framework that can accommodate the unique constraints and demands of hyper-distributed applications, enabling precise control over latency, throughput, reliability, and energy consumption.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Establish measurable SLOs for latency, throughput, availability, reliability, and energy efficiency, with performance testing, continuous monitoring, and stakeholder validation ensuring compliance.		
Dependencies	Monitoring service and telemetry data.		

Requirement ID	F_SO.3	**Priority**	Must-have
Requirement Title	Seamless and declarative orchestration of self-organized distributed orchestration systems.		

Stakeholders	Infrastructure Owners, Service Providers.	**Actors**	ICCS, RYAX, NUBIS
Description	The orchestration system should support distributed control and management, divided into multiple local control loops operating at the Association level while dynamically collaborating. Low-level orchestrators will be responsible for the detailed orchestration of the resources under their control. The high-level orchestrator should follow a declarative approach to describing the deployment objectives at local orchestration mechanisms.		
Rationale/Goal	EMPYREAN distributed and hierarchical orchestration mechanisms should efficiently and autonomously orchestrate services and place data in the heterogeneous and collaborative continuum. Multiple orchestration agents will cooperate and coordinate their operation across Associations to provide the required end-to-end cognitive orchestration, collaborative operation, and self-driven adaptability within the EMPYREAN platform.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Ability to coordinate different orchestration platforms.		
	Seamless deployment in heterogeneous and federated IoT, edge, and cloud resources.		
	Services are efficiently orchestrated in a collaborative continuum of heterogeneous resources.		
Dependencies	Common schemas for describing deployment objectives and end user preferences.		

Requirement ID	F_SO.4	**Priority**	Must-have
Requirement Title	Policy-based orchestration and efficient resource allocation.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	ICCS, RYAX, NUBIS
Description	EMPYREAN should allow the definition of policies for application deployment and data storage that govern the decisions of orchestration mechanisms and the behaviour of deployment mechanisms at each level. EMPYREAN should ensure that these policies are enforced consistently to maintain system objectives such as performance, security, and energy efficiency. This policy-based orchestration will facilitate efficient resource allocation based on current demands and constraints.		
Rationale/Goal	By adopting a policy-based orchestration approach and focusing on efficient resource allocation, EMPYREAN will effectively manage the Association-based IoT-edge-cloud continuum, achieving scalability, high performance, cost efficiency, and robust operational capabilities.		
Relation to UCs	All EMPYREAN use cases.		

Acceptance Measures	Provide flexibility to changing workloads and infrastructure conditions through dynamic policy adjustments.
	Enforce security policies to protect data and assure controlled access.
	Ensure that applications and services can operate seamlessly across Associations.
Dependencies	Comprehensive monitoring and analytics mechanisms.
	Automated and data-driven orchestration tools.
	Interoperable description of deployment policies.

Requirement ID	F_SO.5	**Priority**	Must-have
Requirement Title	Context awareness and autonomous adaptive response.		
Stakeholders	Service Providers, Application Developers	**Actors**	ICCS, RYAX, NEC, UMU
Description	Service orchestration mechanisms should leverage contextual information to make informed and adaptive decisions. This will ensure that resources are utilized optimally, aligning with current conditions and stakeholders' preferences and needs. Moreover, context awareness will enable the orchestration layer to understand and respond to changes in the continuum's highly dynamic environment, optimizing performance and resource usage both within and across Associations.		
Rationale/Goal	By implementing these functionalities, the EMPYREAN orchestration mechanisms will be able to apply recovery actions provided by the service assurance mechanisms as well as adaptations generated by the decision-making mechanisms to optimize the deployed workloads, available data, and operation of Associations.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	React to rapid changes in computational and data demands.		
	Maximize the number of served demands considering their deployment objectives and available infrastructure resources.		
	Apply suggested deployment optimization and recovery actions.		
Dependencies	Support for application-level adaptations.		
	Mechanisms to collect context data from various IoT devices, sensors, and edge nodes.		

Requirement ID	F_SO.6	**Priority**	Must-have
Requirement Title	Transparent lifecycle management of hyper-distributed application components.		

Stakeholders	Service Providers, Application Developers	**Actors**	ICCS, RYAX, NUBIS
Description	The EMPYREAN platform orchestration mechanisms must facilitate the complete lifecycle management for all microservices and application components, regardless of the underlying infrastructure resources (e.g, IoT devices, edge, cloud). The platform should offer infrastructure-agnostic interfaces to support the deployment, update, migration, and termination of any hyper-distributed application that complies with the deployment procedures of the EMPYREAN platform.		
Rationale/Goal	This design requirement guarantees seamless and efficient management of applications across diverse environments. It will facilitate the deployment and management of novel hyper-distributed applications across an Association-based continuum. Additionally, it will allow end uses to review the status of all the application components.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Infrastructure-agnostic deployment for cloud-native applications.		
	Autonomous and continuous monitoring of application components.		
	Minimize manual interventions for application updates and redeployments.		
Dependencies	Automation and orchestration tools.		
	Common schemas for describing deployment objectives and end user preferences.		

Requirement ID	F_SO.7	**Priority**	Must-have
Requirement Title	Coordinate workload migration within and across Associations.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	ICCS, NUBIS
Description	The EMPYREAN orchestration mechanisms must be able to either pause, move, and restart an ongoing workload, or should be able to re-deploy the same workload in another resource in the same or other Association. These capabilities are essential to provide collaborative operation and self-driven adaptability.		
Rationale/Goal	The EMPYREAN service assurance mechanisms based on machine reasoning techniques will automatically trigger the adaptation of the deployed applications to ensure that applications' operational requirements are always satisfied and to provide self-driven adaptability for enhanced fault tolerance and load balancing.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Workload resumes its operation without user intervention.		
	Imposed delay and downtime are acceptable.		

Dependencies	Service assurance and resource orchestration mechanisms.
	Capabilities of underlying IoT-edge-cloud platforms.
	Cross-platform container-based deployment mechanisms.

Requirement ID	F_SO.8	**Priority**	Must-have
Requirement Title	Support automatic data migration operations within and across Associations.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	ICCS, CC, ZSCALE
Description	The EMPYREAN service orchestration mechanisms must support the definition of data placement constraints, which should be considered in decision-making processes for application deployment. The platform should ensure the required data is always available to the deployed hyper-distributed workloads. Additionally, orchestration mechanisms across various Associations should collaborate and coordinate the required operations.		
Rationale/Goal	Orchestration mechanisms should automatically and seamlessly migrate data within and across Associations. These data movements will be initiated by the EMPYREAN platform's decentralized intelligence mechanisms to balance the workload, adapt to changes in available resources, and mitigate any issues that hinder overall performance.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Applications' high-level requirements and operation constraints are met.		
	All required data are automatically transferred to the new locations.		
Dependencies	Secure storage service.		
	Data distributor.		
	Network and storage resources telemetry information.		

Requirement ID	F_SO.9	**Priority**	Must-have
Requirement Title	Implementation and integration of custom scheduling policies.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	ICCS, RYAX
Description	EMPYREAN orchestration mechanisms must leverage cognitive orchestration and multi-agent speculative policies for managing Associations and assigning application workloads and data. Novel resource allocations algorithms will be developed and integrated incrementally across the orchestration mechanisms at various layers (i.e., Decision Engine, Local Schedulers). Thus, the orchestration components should be designed for extensibility, allowing for the seamless integration of new policies based on a well-defined procedure without hidden interfaces.		

Rationale/Goal	The EMPYREAN orchestration mechanisms at both the Aggregator level (e.g., EMPYREAN Orchestrator) and the platform level (e.g., K8s, K3s) will be designed to support the seamless integration of new and advanced scheduling algorithms. This extensibility will ensure that modifications and enhancements can be made without impacting other platform components. This approach ensures transparency and facilitates continuous enhancement, enabling the platform to adapt to evolving requirements and leverage advanced orchestration capabilities.
Relation to UCs	All EMPYREAN use cases.
Acceptance Measures	Well-defined interfaces to describe scheduling policies, input parameters and results.
	Extend the library of scheduling policies without breaking the implementation and functionality of other orchestration components.
Dependencies	Orchestration platforms at the individual edge and cloud infrastructures.
	Application and resource description schemas.

Requirement ID	F_SO.10	**Priority**	Must-have
Requirement Title	Seamless orchestration and management of both container-based and serverless workloads.		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	ICCS, RYAX, NUBIS
Description	EMPYREAN platform should handle the deployment and operation of these different types of workloads by providing an abstraction layer for managing both containers and serverless functions. These mechanisms aim to improve operational efficiency, enhance developer productivity, and improve flexibility and agility of hyper-distributed applications. This approach will support application-level adaptations without changing the deployment descriptor or the application logic while maintaining the benefits of the cloud-native ecosystem.		
Rationale/Goal	EMPYREAN aims to unify the management processes for these two main deployment modes, ensuring that applications' workloads are deployed optimally regardless of their underlying architecture. Implementing this seamless functionality can facilitate a robust, flexible, and cost-efficient operational framework that can support modern hyper-distributed application requirements.		
Relation to UCs	All EMPYREAN use cases.		
Acceptance Measures	Support elasticity and scalability of hyper-distributed applications.		
	Successful deployment of workloads in both deployment modes.		
	Application packaging for seamless deployment of functions.		

Dependencies	Open Container Initiative (OCI) compatible container runtime environment.

Requirement ID	F_SO.11	**Priority**	Must-have
Requirement Title	Flexible Hardware-Accelerated Execution		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	NUBIS
Description	Enable flexible execution by mapping hardware-accelerate-able workloads to relevant hardware functions, decoupling applications from hardware-specific code. This is crucial for real-time, high-frequency data processing in robotic machining cells (UC1) and compute-intensive tasks in logistics operations (UC3). For example, real-time anomaly detection in machining requires processing high-frequency data streams, and leveraging hardware acceleration can significantly enhance performance and reduce latency. In logistics, robots performing order-picking can benefit from offloading intensive computations to nearby devices, improving operational efficiency and response times.		
Rationale/Goal	To ensure applications can leverage hardware acceleration without being tied to specific hardware, enhancing portability and efficiency.		
Relation to UCs	UC1, UC3		
Acceptance Measures	Successful execution of hardware-accelerate-able workloads on at least 3 different types of hardware in real-time anomaly detection and logistics operations.		

Requirement ID	F_SO.12	**Priority**	Must-have
Requirement Title	Offload Acceleration to Nearby Devices		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	NUBIS
Description	Allow IoT devices to request compute-intensive tasks to be executed by an available neighboring node. This will enhance performance and efficiency in IoT networks used in logistics operations (UC3). For example, logistics robots can offload heavy computational tasks such as path planning or object recognition to nearby more powerful devices, thus conserving their own processing power for real-time operations and improving overall system efficiency.		
Rationale/Goal	To utilize remote hardware accelerators, enhancing performance and efficiency in IoT networks. Offloading tasks to nearby devices reduces latency and improves the responsiveness of IoT applications.		
Relation to UCs	UC3		

Acceptance Measures	Successful offloading of tasks from at least 2 types of IoT devices to more than 3 types of edge devices.

Requirement ID	F_SO.13	**Priority**	Must-have
Requirement Title	OCI-Compatible Container Images		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	NUBIS
Description	Bundle binary artifacts along with their descriptors into OCI-compatible container images to facilitate deployment of sensing applications in agricultural fields (UC2) and robotic machining cells (UC1). Using OCI-compatible containers ensures that applications are portable and can run consistently across various environments, which is crucial for maintaining operational consistency in dynamic and heterogeneous settings such as fields and manufacturing plants.		
Rationale/Goal	To ensure applications can be easily deployed across various computing environments, enhancing interoperability. Standardizing on OCI-compatible containers facilitates seamless application deployment and management.		
Relation to UCs	UC1, UC2, UC3		
Acceptance Measures	Creation of OCI artifacts bootable on at least 3 different hardware architectures used in respective environments.		

Requirement ID	F_SO.14	**Priority**	Must-have
Requirement Title	Support Diverse Execution Environments		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	**Actors**	NUBIS
Description	Facilitate the deployment of applications across various execution environments including unikernels and IoT devices. This is crucial for agricultural sensing applications (UC2) and logistics operations (UC3). For example, deploying applications on lightweight unikernels can significantly improve performance and reduce overhead, which is essential for resource-constrained IoT devices in agricultural fields or logistics robots.		
Rationale/Goal	To provide flexibility and compatibility with existing container orchestration ecosystems. Supporting diverse execution environments ensures that applications can be optimized for performance and resource efficiency.		
Relation to UCs	UC2, UC3		
Acceptance Measures	Successful deployment of applications in at least 3 different execution environments (containers, sandbox containers and unikernels) in respective settings.		

Requirement ID	F_SO.15	Priority	Must-have
Requirement Title	Reproducible Environment Packaging		
Stakeholders	Infrastructure Owners, Service Providers, Application Developers	Actors	RYAX, NUBIS
Description	EMPYREAN should provide ways to perform reproducible environment packaging for different types of environments containers or unikernels and specifically build different flavours of the environments depending of the usage context. Furthermore, the packaging needs to be reproducible, lightweight and efficient in order to take into account the possible deployment of environments in different computation demanding contexts.		
Rationale/Goal	The motivation of this requirement is to provide common, reproducible techniques to build different types of standardized environments containers or unikernels.		
Relation to UCs	All		
Acceptance Measures	Ensure successful, reproducible deployment of applications across at least 3 standardized environments, with performance, consistency, and stakeholder satisfaction validated through automated processes, comprehensive documentation, and compliance with security standards.		

6.1.7 Integration and Platform Development Requirements

This section outlines the requirements for Platform Integration and Testing, which are essential for unifying the outcomes of the developed components and services into the integrated EMPYREAN platform. These requirements will support the development activities of EMPYREAN components and enhance the overall integrated performance. Specifically, they will provide guidelines for the technological developments in WP3 and WP4, and support each UC through adaptation, customization, and further development of UC leaders' existing services and applications. Additionally, these requirements ensure the adoption of the DevSecOps application development approach, promoting flexible collaboration between development teams. This approach facilitates the frequent release of integrated versions of the EMPYREAN platform, ensuring continuous improvement and responsiveness to evolving needs.

Table 10: Analysis of integration and platform development requirements

Requirement ID	F_IPDR.1	Priority	Must-have
Requirement Title	Expose well-defined APIs through EMPYREAN SDK.		
Stakeholders	Application Developers	Actors	All

Description	The EMPYREAN platform must provide a comprehensive Service Development Kit (SDK) that includes a set of well-defined APIs. This SDK will enable seamless interaction with the platform's components. In addition, it will facilitate the robust reporting and monitoring of deployed applications and workloads. By abstracting the complexities involved in creating and deploying hyper-distributed cloud-native applications, the EMPYREAN SDK will simplify user operations, eliminating the need for low-level direct interaction with the individual services and resources across the IoT-edge-cloud continuum.
Rationale/Goal	The EMPYREAN SDK should streamline the development of use cases for the final demonstrations and the integration of platform components. It will assist developers in building, deploying, and managing hyper-distributed applications within the EMPYREAN platform. Additionally, it will support the implementation of future extensions and the deployment of diverse cloud-native applications.
Relation to UCs	EMPYREAN use cases will be built upon the robust foundation provided by the EMPYREAN SDK and its exposed APIs, ensuring easy and seamless integration with the platform's technological solutions.
Acceptance Measures	No hidden internal APIs.
	Provide high-level methods to interact with the core platform components (e.g., Service Orchestrator, Decision Engine, Aggregator, Data Distributor, Storage Service, Telemetry Service)
Dependencies	Exposed APIs by the integrated platforms and services.
	Abstraction models for applications and resources.

Requirement ID	F_IPDR.2	**Priority**	Must-have
Requirement Title	Build upon well-established open-source platforms and consortium existing solutions.		
Stakeholders	Infrastructure Owners, Service Providers	**Actors**	All
Description	The EMPYREAN platform must not be built from scratch but should extend and integrate existing open-source technologies, solutions from the project consortium, and potential synergies with other relevant initiatives. It must follow a modular architecture and adopt standardized and open interfaces.		
Rationale/Goal	Preference should be given to expanding existing frameworks, tools, and open standards to enhance openness in the evolving continuum landscape. This way, EMPYREAN will also contribute to the provision of an open edge ecosystem, fostering innovation of the whole European ecosystem.		
Relation to UCs	All EMPYREAN use cases will utilize the provided novel services and mechanisms that will be built upon this strategic approach.		

Acceptance Measures	Maximize the use of EMPYREAN enablers belonging to consortium portfolio.
	Maximize the use of European open-source technologies and frameworks.
Dependencies	Availability and portability of available solutions regarding the target system architectures and programming languages.
	Detailed workflows and interface documentation for the EMPYREAN-developed components.

Requirement ID	F_IPDR.3	**Priority**	Must-have
Requirement Title	Documentation of all integration points.		
Stakeholders	Application Developers, Service Providers	**Actors**	All
Description	All the integration points should be well documented. Integration points refer to components or modules that will be part of or use the EMPYREAN platform. The documentation should cover cases of successful and unsuccessful communication.		
Rationale/Goal	Documentation will facilitate discovery of EMPYREAN components' features by third-parties, guaranteeing a minimum level of interoperability and portability.		
Relation to UCs	By documenting the integration points, the EMPYREAN platform can ensure smooth, efficient, and reliable CI/CD processes that support the successful implementation of project use cases.		
Acceptance Measures	Documentation should cover all relevant endpoints adhering to well-known specification (e.g., OpenAPI v3.0).		
	Correctly implemented integration points that work as described in the documentation.		
	Documented integration points should be known and agreed with involved members that use it.		
Dependencies	EMPYREAN architecture describing all integration points of the system, as will be provided by D2.2 (M7) and D2.3 deliverables (M12) and the components implementation within WP3 and WP4 that will be delivered in two versions (M15 and M30).		

Requirement ID	F_IPDR.4	**Priority**	Must-have
Requirement Title	Docker image of developed components for creating running containers.		
Stakeholders	Application Developers, Service Providers	**Actors**	All
Description	For every platform component, there should be at least one Docker image to be used for deploying this component in a container that will integrate		

	with other components. In case multiple containers are required, there should be a docker-compose file.
Rationale/Goal	Components running in containers greatly simplify the integration process. This methodology allows the CI/CD process to deploy more than one component on the same machine without unexpected conflicts.
Relation to UCs	Implementing robust CI/CD pipelines ensures rapid, reliable, and repeatable software delivery, which is essential for the success of EMPYREAN use cases.
Acceptance Measures	One Dockerfile or one image per component is provided.
	The container can run without issues. No errors should exist in the container logs, warnings can be present when not affecting the execution.
	Provision of required configuration parameters (e.g, ports, environment variables).
Dependencies	A functional component already developed and documented.

Requirement ID	F_IPDR.5	**Priority**	Should-have
Requirement Title	EMPYREAN Git-based code repository.		
Stakeholders	Application Developers	**Actors**	All
Description	A Git-based code repository (e.g., GitHub, GitLab) will be used in EMPYREAN for all developed EMPYREAN components. The code should be placed in the corresponding work package and partner folder.		
Rationale/Goal	CI/CD pipelines should be able to retrieve available code from a shared location. The most popular Git-based online repositories also integrate with CI/CD components (e.g., Jenkins) that will build the components.		
Relation to UCs	Implementing robust CI/CD pipelines ensures rapid, reliable, and repeatable software delivery of, which is essential for the success of EMPYREAN UCs.		
Acceptance Measures	Component code placed into the corresponding Gitlab folder.		
	Readme file or script that uses this code to build the component.		
Dependencies	Repository account for every user.		

Requirement ID	F_IPDR.6	**Priority**	Should-have
Requirement Title	CI/CD guidelines.		
Stakeholders	Application Developers, Service Providers	**Actors**	All

Description	All partners should follow the provided CI/CD guidelines. Code development must follow the guidelines and methodology imposed by the CI/CD pipeline and DevSecOps approach.
Rationale/Goal	EMPYREAN will adopt a DevSecOps approach to continuously integrate the developed mechanisms and frameworks. As part of the CI/CD process, code development should follow good practices, as described in the DevSecOps approach.
Relation to UCs	Implementing robust CI/CD pipelines ensures rapid, reliable, and repeatable software delivery, which is essential for the success of EMPYREAN use cases.
Acceptance Measures	Steps of CI/CD pipeline should be defined in a Jenkins file.
	Automated development, testing, and deployment of the EMPYREAN platform core components.
Dependencies	Well defined system platform architecture.
	Collaboration tools to ensure smooth CI/CD processes.

Non-functional Requirements

The non-functional requirements express quality requirements that the components of the implemented platform should satisfy. For the non-functional requirements, EMPYREAN adopts the widely accepted ISO/IEC 25010[148] standard model of quality characteristics, which determines the quality characteristics that must be considered when evaluating the properties of a software product. A system's quality is the degree to which it satisfies its various stakeholders stated and implied needs and thus provides value. The ISO/IEC25010quality model classifies software quality in a structured set of characteristics and sub-characteristics. Next, we briefly describe the EMPYREAN platform ecosystem's relevant characteristics for each non-functional requirement.

Table 11: Analysis of overall non-functional requirements

Requirement ID:	NF_GR.1	**Priority:**	Core
Requirement Title:	Performance Efficiency	**Stakeholders Involved:**	All

[148] ISO/IEC, "ISO/IEC 25010: 2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, (2011).

Description:	This characteristic represents the performance relative to the number of resources used under stated conditions. This characteristic is composed of the following sub-characteristics: ● **Time behaviour** - Degree to which the response, processing times and throughput rates of a product or system, when performing its functions, meet the requirements. ● **Resource utilization** - Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet the requirements. ● **Capacity** - Degree to which the maximum limits of a product or system parameter meet requirements.
Rationale / Goal:	The EMPYREAN platform integrates the project's developed technologies related to trust, security, and seamless data, computing and application management, coupling them with intelligent mechanisms to optimize resource utilization. EMPYREAN will be able to dynamically cater to application and user constraints while calibrating the configuration of the available resources.

Requirement ID:	NF_GR.2	**Priority:**	Core
Requirement Title:	Functional Suitability	**Stakeholders Involved:**	All
Description:	This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is composed of the following sub-characteristics: ● **Functional completeness** - Degree to which the set of functions covers all the specified tasks and user objectives. ● **Functional correctness** - Degree to which a product or system provides the correct results with the needed degree of precision. ● **Functional appropriateness** - Degree to which the functions facilitate the accomplishment of specified tasks and objectives**.**		
Rationale / Goal:	The EMPYREAN platform will provide an abstraction layer that automates the application deployment by continuously optimizing the allocation and configuration of the available heterogeneous resources. Moreover, it will support the autonomous adaptation and management of the deployed services and resources to ensure that applications perform as intended.		

Requirement ID:	NF_GR.3	**Priority:**	Core
Requirement Title:	Compatibility	**Stakeholders Involved:**	All

Description:	Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment. This characteristic is composed of the following sub-characteristics: ● **Co-existence -** Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product. ● **Interoperability** - Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
Rationale / Goal:	The EMPYREAN platform integrates decentralized and heterogeneous IoT, edge, and cloud resources through an Association-based paradigm. EMPYREAN will build the appropriate abstractions and system-level mechanisms to promote resource interoperability and seamless application deployment. The platform will not be built from scratch but as an integration of available well-known solutions by implementing all the required extensions and improvements.

Requirement ID:	NF_GR.4	**Priority:**	Core
Requirement Title:	Usability	**Stakeholders Involved:**	All
Description:	Degree to which a product or system can be used by specified users to achieve specific goals with effectiveness and efficiency in a specified context of use. This characteristic is composed of the following sub-characteristics: ● **Appropriateness recognizability** - Degree to which users can recognize whether a product or system is appropriate to their needs. ● **Learnability -** Degree to which a product or system can be used by specified users to achieve specific goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use. ● **Operability** - Degree to which a product or system has attributes that make it easy to operate and control. ● **User error protection** - Degree to which a system protects users against making errors.		
Rationale / Goal:	EMPYREAN will support dynamic application deployment and adaptation, as well as elasticity and scalability of hyper-distributed applications, by adopting a develop once, deploy everywhere approach. Users will develop their applications and express their high-level deployment requirements in an infrastructure-agnostic manner. EMPYREAN will provide the appropriate technological solutions to facilitate developing, packaging, deploying, and monitoring hyper-distributed applications.		

Requirement ID:	NF_GR.5	**Priority:**	Core
Requirement Title:	Reliability	**Stakeholders Involved:**	All
Description:	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub-characteristics: ● **Maturity** - Degree to which a system, product or component meets needs for reliability under normal operation. ● **Availability** - Degree to which a system, product or component is operational and accessible when required for use. ● **Fault tolerance** - Degree to which a system, product or component operates as intended despite the presence of hardware or software faults. ● **Recoverability** - Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.		
Rationale / Goal:	EMPYREAN will enable application-level adaptation and scalability for hyper-distributed applications through AI-driven service assurance and autoscaling mechanisms. These advancements will ensure the availability, reliability, and efficiency of deployed applications by continuously monitoring them and initiating necessary self-optimization procedures automatically.		

Requirement ID:	NF_GR.6	**Priority:**	Core
Requirement Title:	Security	**Stakeholders Involved:**	All
Description:	Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. This characteristic is composed of the following sub-characteristics: ● **Confidentiality** - Degree to which a product or system ensures that data are accessible only to those authorized to have access. ● **Integrity** - Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data. ● **Non-repudiation** - Degree to which actions or events can be proven to have taken place so that events or actions cannot be repudiated later. ● **Accountability** - Degree to which the actions of an entity can be traced uniquely to the entity. ● **Authenticity** - Degree to which the identity of a subject or resource can be proved to be the one claimed.		

Rationale / Goal:	The EMPYREAN platform will encompass several security, trustworthiness, and cyber threat intelligence mechanisms, operating in multiple layers, to provide by design, privacy and resiliency against security threats.

Requirement ID:	NF_GR.7	**Priority:**	Core
Requirement Title:	Maintainability	**Stakeholders Involved:**	All
Description:	This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. This characteristic is composed of the following sub-characteristics: ● **Modularity** - Degree to which a system or computer program is composed of discrete components, such that a change to one component has minimal impact on other components. ● **Reusability** - Degree to which an asset can be used in more than one system, or in building other assets. ● **Analysability** - Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified**.** ● **Modifiability** - Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality. ● **Testability** - Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.		
Rationale / Goal:	The integration of the EMPYREAN platform will adhere to DevSecOps best practices. This process will be conducted incrementally according to a structured plan, with local interoperability tests performed after each phase. This meticulous procedure will validate the platform's technical conformity and efficiency. Furthermore, the chosen modular architecture, along with well-defined and open interfaces, will facilitate the seamless addition of future extensions for individual components.		

Requirement ID:	NF_GR.8	**Priority:**	Core
Requirement Title:	Portability	**Stakeholders Involved:**	All
Description:	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub-characteristics:		

	• **Adaptability** - Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments. • **Installability** - Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment. • **Replaceability** - Degree to which a product can replace another specified software product for the same purpose in the same environment.
Rationale / Goal:	The EMPYREAN platform aims to provide the necessary mechanisms to enable dynamic application deployment and autonomous adaptation over heterogeneous infrastructures. The platform will leverage available open-source technologies and state-of-the-art platforms. Components will interact through well-defined interfaces to ensure future extensions and portability to new hardware and software platforms.

7 Conclusions

The EMPYREAN project sets a new benchmark in the IoT-edge-cloud continuum by addressing critical challenges in distributed trust, secure data management, and AI-driven resource optimization. By integrating advanced technologies such as blockchain, AI, and unikernels, EMPYREAN aims to create a resilient, scalable, and secure infrastructure. Based on this innovative approach, EMPYREAN envisions a new paradigm for the continuum, introducing the concept of collaborative collectives of IoT devices, robots, and resources spanning from the edge to the cloud.

The project's innovative solutions, including a distributed trust management framework and enhanced AI-based workload autoscaling, will significantly enhance the performance and reliability of hyper-distributed applications. Through its comprehensive and forward-thinking approach, EMPYREAN is poised to lead the next generation of IoT-edge-cloud ecosystems, providing substantial benefits across multiple industrial domains.

This deliverable encapsulates the project's thorough analysis of the current state of technologies, an extensive use case analysis, and the subsequent identification of platform requirements and associated KPIs. It includes EMPYREAN's comprehensive evaluation of cutting-edge and beyond state-of-the-art technologies that the project aims to advance. This deliverable also details the EMPYREAN use cases and provides an initial analysis of the functional and non-functional requirements for EMPYREAN components and their associated KPIs.

EMPYREAN requirements are systematically organized to facilitate efficient processing and transformation into architectural decisions. Based on EMPYREAN's goals and technical objectives, the requirements are grouped into seven categories, encompassing both the project use case requirements and the innovative technologies envisioned to enable a cognitive continuum. This continuum integrates intelligence and automation to achieve more efficient data processing.

Finally, the deliverable describes the project's relation to ongoing relevant EU projects in similar areas.