D5.7 Pilots operation and monitoring III

Document Identification			
Status	Review	**Due Date**	30/11/2022
Version	1.0	**Submission Date**	30/11/2022

Related WP	WP5	**Document Reference**	D5.7
Related Deliverable(s)	D5.3, D5.4	**Dissemination Level (*)**	PU
Lead Participant	FILL	**Lead Author**	Verena Stanzl
Contributors	I2CAT, HOLO, ATOS, ICCS, ENG, INTRA, INNOV, IMI	**Reviewers**	Panagiotis Matzakos (INTRA)
			Gabriele Giammatteo (ENG)

Keywords:
Pilots Operation, Pilots Monitoring, Metrics, Experimentation

Document Information

List of Contributors	
Name	Partner
Verena Stanzl	FILL
Michael Gillhofer	FILL
Tobias Gann	FILL
Stefan Stockhammer	FILL
August Betzler	i2CAT
Estela Carmona	i2CAT
Nour Fendri	HOLO
Carina Pamminger	HOLO
Alexander Werlberger	HOLO
Clayton Gordy	HOLO

Document History			
Version	Date	Change editors	Changes
0.0	19/10/2022	Verena Stanzl (FILL)	TOC established and responsible partners assigned
0.1	4/11/2022	Verena Stanzl (FILL), August Betzler (i2CAT), Estela Carmona (i2CAT)	Inputs of UC2 and UC3 integrated, Section 7 integrated, Introduction added, Section 3 added
0.2	09/11/2022	Verena Stanzl (FILL), Nour Fendri (HOLO), Clayton Gordy (HOLO)	Inputs of UC1 integrated
0.3	09/11/2022	Verena Stanzl (FILL)	Section 3 added, Executive Summary, Conclusions added
0.4	14/11/2022	Verena Stanzl (FILL)	Version for internal review
0.5	22/11/2022	Verena Stanzl (FILL), Panagiotis Matzakos (INTRA), Gabriele Giammatteo (ENG)	Version with included feedback, comments and corrections
0.6	23/11/2022	Verena Stanzl (FILL), Estela Carmona (i2CAT)	Addressed comments and corrections in general Sections (1, 2, 3, 8), updated Sections 5 and 6.
0.7	24/11/2022	Verena Stanzl (FILL), Clayton Gordy (HOLO)	Updated Section 4 with revised inputs from UC1
0.8	25/11/2022	Verena Stanzl (FILL)	Version for quality control
0.9	25/11/2022	Carmen San Román (ATOS)	Quality assurance check
1.0	28/11/2022	Lara López (ATOS)	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Verena Stanzl (FILL)	25/11/2022
Quality manager	Carmen San Román (ATOS)	25/11/2022
Project Coordinator	Lara López (ATOS)	28/11/2022

Table of Contents

Document Information 2
Table of Contents 4
List of Tables 6
List of Figures 7
List of Acronyms 8
Executive Summary 9
1 Introduction 10
1.1 Purpose of the document 10
1.2 Relation to other project work 10
1.3 Structure of the document 10
2 Overview of the work performed in this period (M30 – M36) 11
3 Overview of Pledger components 12
4 Use Case 1: Mixed Reality applications on the edge 14
4.1 Integration with Pledger 15
4.2 Pilot deployment 16
4.2.1 AR hardware/software setup requirements 17
4.2.2 Pledger Toolkit Setup Requirements 17
4.2.3 Non-technical Setup Requirements 17
4.3 Experiments performed 18
4.3.1 Fast Prototyping 18
4.3.2 Remote Manufacturing and Service Support & Training Interactions 18
4.4 Feedback on using Pledger 19
5 Use Case 2: Edge infrastructure for enhancing safety of vulnerable road users 20
5.1 Integration with Pledger 21
5.1.1 SOE Framework – RAN Controller 21
5.1.2 RDNS – DLT (storing data on the blockchain) 22
5.1.3 RDNS – Monitoring Engine 23
5.1.4 RDNS – DLT (Smart Contracts) 23
5.2 Pilot deployment 24
5.3 Experiments performed 26
5.3.1 Slice creation and service deployment 27
5.3.2 Risk detection and Notification System 27
5.3.3 Showcasing QoE assurance and DLT features 28
5.4 Feedback on using Pledger 28
6 Use Case 3: Manufacturing the data mining on the edge 30

6.1 Integration with Pledger 30
6.1.1 Connectivity 30
6.1.2 UC Application – Monitoring Engine 31
6.1.3 UC Application - DLT 31
6.1.4 UC Application - StreamHandler 32
6.1.5 Smart contracts in UC3 32
6.2 Pilot deployment 33
6.3 Experiments performed 34
6.3.1 Service deployment and monitoring performance 34
6.3.2 Warm-up of the machine and manufacturing 35
6.3.3 Showcasing QoE, SLAs and improvement of performance 35
6.3.4 Accessing sensitive information on the DLT 36
6.4 Feedback on using Pledger 36
7 Multi Mobile Network Operator PoC 38
7.1 Integration with Pledger 39
7.2 Experiments performed 39
8 Conclusions 41
9 References 42
Annexes 43

List of Tables

Table 1: Status integration endpoints UC1 16
Table 2: Status integration endpoints UC2 24
Table 3: Status integration endpoints UC3 32

List of Figures

Figure 1: Connected Pledger infrastructure 13
Figure 2: Schematic depicting the integration of the PledgAR Workspace with the Pledger toolkit 15
Figure 3: Snippet from a log showing an OBU connected to the application (station_id 2). 22
Figure 4: Pledger testbed in Barcelona with the main DC at the i2CAT premises in Zona Universitaria, as well as the edge and on-street deployment around the Fluvià tram station. 25
Figure 5: The layout of Fluvià tram station (and other stations in Barcelona), where the bicycle lane (middle), separates the tram stop area from the pedestrian area. 25
Figure 6: V2X Slice Diagram 26
Figure 7: Example of UC3 metrics on StreamHandler 31
Figure 8: Pilot machine in the Fill Future Zone 34
Figure 9: Visualization of the thermal stability indicator evolvement 35
Figure 10: Design of the proof of concept to validate the 5G integration with Pledger. 38

List of Acronyms

Abbreviation / acronym	Description
3D	Three-Dimensional
AR	Augmented Reality
CAD	Computer-Aided-Designs
CAM	Cooperative Awareness Message
DENM	Decentralized Environmental Notification Message
DLT	Distributed Ledger Technology
DSS	Decision Support System
Dx.y	Deliverable number y belonging to WP x
GNSS	Global Navigation Satellite Systems
ISAR	Interacting Streaming for Augmented Reality
JSON	JavaScript Object Notation
K8s	Kubernetes
MNO	Mobile Network Operator
MOCN	Multi-Operator Core Networks
MQTT	Message Queuing Telemetry Transport
MR	Mixed Reality
NR	New Radio
OBU	On-Board Unit
OSM	Open Source Management and Orchestration
PC	Personal Computer
PLC	Programmatic Logical Control
PoC	Proof of Concept
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RDNS	Risk Detection and Notification Service
SDK	Software Development Kit
SLA	Service-Level Agreement
UC	Use Case
UI	User Interface
V2X	Vehicle-to-Everything
V2XCOM	V2X-Communication
VM	Virtual Machine
VPN	Virtual Private Network
VRU	Vulnerable Road User
WebRTC	Web Real-Time Communication
WP	Work Package

Executive Summary

Task T5.3 “Pilots operation and monitoring” handles the activities for the on-site preparations of the demonstrators’ deployment and pilots’ set-up to validate the Pledger Use Cases (UCs). The task consists of three iterations. In the first iteration, the infrastructure was set-up for the UCs and the first experimentation scenarios were outlined. The second iteration documents the implementation and execution of these scenarios.

In this deliverable, the third iteration of this task is documented. It focuses on the actual pilot deployment in the relevant environment and the execution of experiments and scenarios to validate the Pledger components and the benefits they enable for the UCs. End users are involved to gather the feedback and evaluate the pilot KPIs which will be documented in the deliverable D5.8 “Pledger overall validation and evaluation” [8]. However, the experiments to gather and achieve these results are reported in this document. Furthermore, feedback for using Pledger and its components of the internal users (developers of the UC) is gathered and reported in this document.

The execution of experiments focused – similar to the previous iteration – on exploiting and demonstrating the functionalities offered by Pledger system: the reservation of dedicated computation and network resources, the orchestration of different services, the monitoring of their performance and their associated Service Level Agreements (SLAs) as well as the automated execution of performance optimization based on decision-making algorithms. These functionalities are extended with the use of the Distributed Ledger Technology (DLT) to manage sensitive data and the implementation of smart contracts.

The UC1 presents the pilot deployment in an industrial setting with integration of virtual content over Augmented Reality (AR) solutions for collaboration, remote support and training procedures. The UC benefits from Pledger via orchestration and management of computational resources, monitoring metrics and a continual maintenance of a high level of data security.

The UC2 presents the on-street pilot deployment in Barcelona city infrastructure and the use of applications to notify end users about potential risks in the public space. The experiments to showcase the risk detection and feedback gathering mechanisms are described. This UC benefits especially from the automated resource reservation and slice creation, the functionalities of the DLT and monitoring services.

The UC3 presents the actual pilot deployment in the manufacturing environment. The UC benefits from orchestrating services, monitoring the performance and storing sensitive information in a safe way. Furthermore, scenarios to showcase the Pledger functionalities to ensure no unexpected interruptions during the pilot are presented.

Lastly, Pledger also supports 5G, which is demonstrated with the Multi Mobile Network Operator Proof-of-concept in the Pledger indoor lab in Barcelona. In this deliverable, the execution of the experiments is described, and the benefits of full automatized deployment are demonstrated.

All results achieved and obtained through the pilot executions are used for the Pledger evaluation and are described in the associated deliverable D5.8 [8].

1 Introduction

1.1 Purpose of the document

Task T5.3 “Pilots operation and monitoring” handles the activities for the on-site preparations of the demonstrators’ deployment and pilots’ set-up to validate the Pledger use cases.

The task consists of three iterations and in this deliverable, the work performed and results achieved in the third and last iteration of this task are documented.

In the first deliverable of this task, D5.3 [1], the infrastructure set-up for the UCs was described and as well as the first experimentation scenarios were outlined. In the second iteration of this task, these scenarios were implemented and performed, and the achieved results and observations are documented in D5.4 [2].

In this iteration of the task, the actual pilot deployment is in focus and the experiments performed during the pilot are in focus. Furthermore, feedback on using Pledger was gathered and documented.

1.2 Relation to other project work

This deliverable builds on the deliverables D5.3 [1] and D5.4 [2] reporting the previous iterations, where the set-up of infrastructure, the integration endpoints and the first scenarios have been reported.

The task focuses on the integration of the UCs with the Pledger Core system and the execution of the pilots and feedback gathering. The integration of the different Pledger components is part of Task T5.2 “Integration and Demonstrators setup” and detailed in the associated deliverables D5.2 [3] and D5.6 [4]. The applications for the UC were developed in the Task T5.1 “Application development for the Use Cases” and documented in D5.1 [5] and D5.5 [6].

The evaluation of the Pledger system and the UCs is handled in Task T5.4 “Validation and evaluation” and documented in the deliverable D5.8 [8].

1.3 Structure of the document

This document is structured in 7 major chapters

Chapter 2 presents an overview of the work performed in this iteration to place it in its overall context.

Chapter 3 recaps the involved Pledger components and their functionality for the UCs.

Chapters 4, 5 and 6 present the work performed for UC1, UC2 and UC3 respectively. A short overview of each pilot is given, followed by a description of the integration with the Pledger Core system. Afterwards, the focus is set on the pilot deployment, the experiments performed in the pilot and the results and outcomes observed. Every UC-chapter includes at the end the feedback on using the Pledger system.

Chapter 7 describes a Proof of Concept (PoC) for a multi mobile network operator, showcasing and demonstrating the ability to deploy 5G network slices.

Chapter 8 closes this deliverable with a conclusion.

2 Overview of the work performed in this period (M30 – M36)

The main efforts during M30 and M36 (June 2022 and November 2022, respectively), regarding pilots' operation consists of the following aspects:

- Establishment of the remaining integration endpoints;
- Preparation of experimentation scenarios in the pilot environment;
- Preparation of surveys for end users in the experiments;
- Preparation of surveys for UC developers to gather feedback about the Pledger components;
- Execution of the pilots;
- Gathering of survey results and demo material.

To enable the final pilot execution, preparatory steps were performed in the previous two iterations. The first pilot iteration was focused on the infrastructure set-up and identification of integration endpoints with Pledger and the first implementation of them. The results and work done in this iteration is documented in D5.3 [1]. In the second iteration, further integration was established, and the first scenarios related to service orchestration, infrastructure provisioning and offloading/scaling resources using Pledger components were demonstrated and documented in D5.4 [2].

In this iteration, the actual pilot execution and demonstration on relevant environment took place. The pilots were set-up, the remaining integration endpoints were established and experiments in the pilots were performed. All three UCs were able to perform the outlined scenarios in the pilot environment.

The pilot execution especially showcases the applications developed in the UCs in a relevant environment., which specifically, are focused on an industrial environment for AR and manufacturing, as well as the public on-street spaces on-street in Barcelona to increase the safety of vulnerable road users. Furthermore, the scenarios integrated Pledger functionalities and demonstrate the benefits end-to-end.

Moreover, surveys to be distributed among the end users during the experiments were prepared. The surveys are divided into two sections: one having more general questions about Pledger and the other dealing with questions specifically for the UC and the experiences obtained during the pilots. The general questions cover mainly topics about the a-prior knowledge of edge and cloud computing, whether participants understand and see the benefits of using Pledger in the context of the UC and whether the metrics are understandable or not. The questions can be found in Annex A. The results of this survey are part of the overall evaluation and are reported in deliverable D5.8 [8].

Finally, feedback among the developers involved in the project and the development of the UC was gathered. The feedback is mainly related to the use of the components, the results achieved using them, the experiences from the integration with the system. This feedback is summarized in the associated sections about the UCs and pilots.

3 Overview of Pledger components

Before going into details for the different UCs and pilots, a short recap and overview of the Pledger components involved in the different scenarios is given. The architecture of the Pledger Core system, as well as the details for the functionalities of the components are given in the deliverable D2.3 [9]. The details for the individual components and subsystems are given in the associated deliverables of WP3 and WP4. The integration and interaction between the different components are part of Task 5.2 and are documented in its corresponding deliverables D5.2 [3] and D5.6 [4].

The following list summarizes the main components mentioned in this document and a short recap of their core functionality:

- Edge-to-Cloud Orchestrator (E2CO)[1]: management of information related to runtime environment specifications, service orchestration (start, stop, update of applications);
- Benchmarking subsystem: provisioning of performance data of configured infrastructures;
- App Profiler: Manual or automatic profiling of applications;
- Decision Support System (DSS): provisioning of suggestions related to the app orchestration to optimize performance;
- SLA Lite: Creation, management and evaluation of the SLAs associated to the applications running and notification to other components in case of SLA violations;
- ConfService: User Interface (UI) for infrastructure and application configuration;
- Distributed Ledger Technology (DLT): Pledger's blockchain network for supporting the respective smart contracts and transactional operations;
- Slicing and Orchestration Engine (SOE) Framework and Radio Access Network (RAN) controller: configuration and management of end-to-end network slices including both radio, network and compute resources, allowing the deployment and orchestration of services on top of these slices;
- Monitoring Engine: collection of cluster infrastructure and application metrics;
- StreamHandler: a high-performance distributed streaming platform, backbone for integration of core components, used by UC1 and UC3 for integration with the Monitoring Engine and for data transfer for offloading in UC3. Furthermore, it is used in UC2 and UC3 for integration with the Pledger DLT and storing sensitive information there.

The infrastructure set-up for the different pilots and the infrastructure hosting the Pledger core components were described in D5.3 [1]. The connectivity of the different infrastructures is established by OpenVPN [10] connections to habilitate full visibility among the different Pledger infrastructures and allow the communication for the control plane and the data plane.

Within multiple available options available for the connectivity of the infrastructures, the consortium started from the needs and constrains raised by each of the infrastructure owners and used an approach which could be easily adopted in the future by potential additional partners. During the project, a site-to-site OpenVPN configuration was configured with reverse tunnel connection. Using this configuration, only OpenVPN servers on ENG and i2CAT are publicly exposed, considering the hard requirement of UC3 to avoid exposure on public Internet.

With such a configuration in place, PODs running on each Kubernetes (K8s) [11] infrastructures (ENG and i2CAT), containers (FILL) and processes (HOLO) are visible to each other, facilitating the distribution of applications over the different infrastructures according to the DSS decisions and the capabilities of the E2CO to configure micro services connectivity. To complement this setup, Pledger also used StreamHandler to allow service to service communication whenever there are no real-time communication requirements.

[1] Both terms, E2CO and Orchestrator are used interchangeable.

Figure 1 shows the available infrastructure and their connection showing the core system hosted at the ENG-infra and the different UC infrastructures. This is an updated image compared to D5.4 [2]. Since the last iteration of the task, one further node was added, namely UC2-infra-30-5G, which is utilized to demonstrate the Multi Mobile Network Operator PoC described in Section 7. The green nodes show the infrastructure of the partners (ENG, UC partners) and their associated nodes. The ENG-infra hosting the Pledger core system consists of 4 nodes (depicted in orange), UC2-infra-30 consists of 6 nodes (both cloud and edge, visualized in yellow) and the infrastructures of UC1 and UC3 have one edge node respectively (visualized in blue). The detailed description of these infrastructures is described in the Deliverable D5.3 [1].

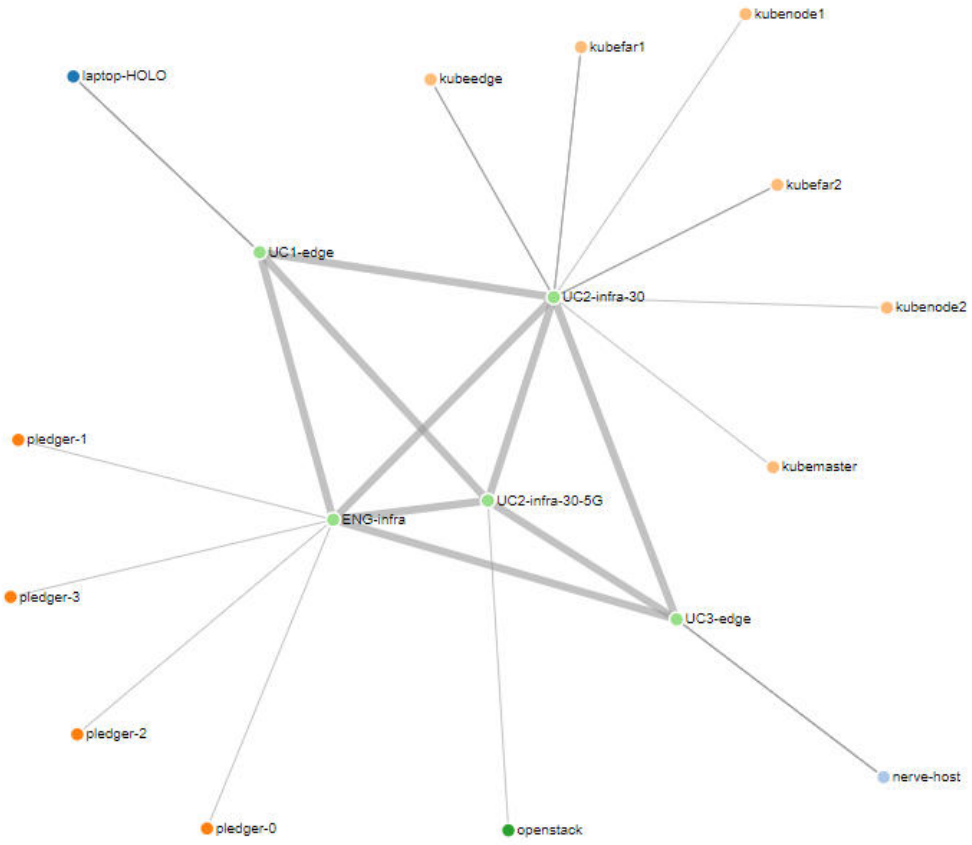

Figure 1: Connected Pledger infrastructure

4 Use Case 1: Mixed Reality applications on the edge

UC1 “Mixed Reality applications on the edge” aims to leverage edge computing technologies provided by the Pledger toolkit to increase the performance of augmented reality (AR) in industrial service solutions. To reach this goal, this UC explores three specific case studies, each one with emphasising on the optimization in of real time mixed reality (MR) interfaces through edge computing infrastructures. Each of the three case studies focused on integrating virtual content in an industrial setting, specifically during:

- Collaboration for Fast Prototyping (1)
- Remote Manufacturing and Service support (2)
- Training procedures (3)

All of these case studies in general necessitate three-dimensional (3D) graphical visualization and manipulation of Computer-Aided Designs (CAD) of various types and polygon size. A necessary requirement in any such 3D hologram visualization is to ensure efficiency and consistency in performance, while simultaneously maintaining a high degree of hologram resolution. This is particularly challenging given that market AR devices are constrained by a lack of computational strength, and thus do not have the ability to perform at high industrial standards on their own. Circumventing this limitation is the ability to offload the computationally heavy algorithms onto an edge device which enables remote rendering of 3D content which can be simply streamed onto the AR device. In doing so, hologram visualization can effortlessly be maintained in these devices with very little loss of service quality, and thus provide optimal conditions for industrial professionals. As detailed in previous iterations of this deliverable, UC1 makes use of the remote rendering software development kit (SDK) Interactive Streaming for Augmented Reality (ISAR) [12] for this purpose. ISAR remote rendering solutions permit end-user experiences to be maintained smoothly (e.g., precise visualization and manipulation ability of holograms) via computational offloading to edge devices.

While ISAR is an optimal solution to the AR edge computing goals illustrated above, dynamic modifications due to changing performance metrics during real life situations must be accounted for. Specifically, speed and performance levels should be able to be maintained consistently in the face of differential business-task demands, to preserve quality of experience (QoE) for relevant users, while also keeping fiscal and environmental impacts in mind. Furthermore, maintaining strict and robust security measures during streaming of confidential trade and industry 3D data is a requisite. Accommodating these latter two requirements was the focus of this Pledger applied use case. UC1 aimed at integrating the Pledger toolkit and architectural platform with the AR edge computing ability provided by ISAR remote rendering services.

Integration of AR components with the Pledger interface is presented for reference below, but in brief it consists of a personal computer (PC) or laptop running a virtual machine (VM) where the AR application for this UC (herein referred to as the PledgAR Workspace) will run. The VM and thus the PledgAR Workspace itself is connected in tandem to the Pledger core infrastructure as well (see below) as peripheral end-user AR devices (HoloLens 2; HL2 [13]).

This deliverable’s iteration will describe the UC1 pilot deployment and experiments performed. The pilot deployment for this UC is done at the on-site premises of FILL. The final results of UC1 demonstrations and experiments are provided in the proceeding D5.8 deliverable [8].

4.1 Integration with Pledger

UC1 integration with the Pledger platform has previously been described in detail in past deliverables, i.e., D5.3 [1] and D5.4 [2]. Here, only an overview of the major components and key functional connectivities will be provided. Several specific integrations with the Pledger core platform are used in UC1. Each integration provides a specific functionality solution which focuses, respectively, on 1) orchestration and management of computational resources,2) monitoring and decision making, 3) high level of data security (Figure 2).

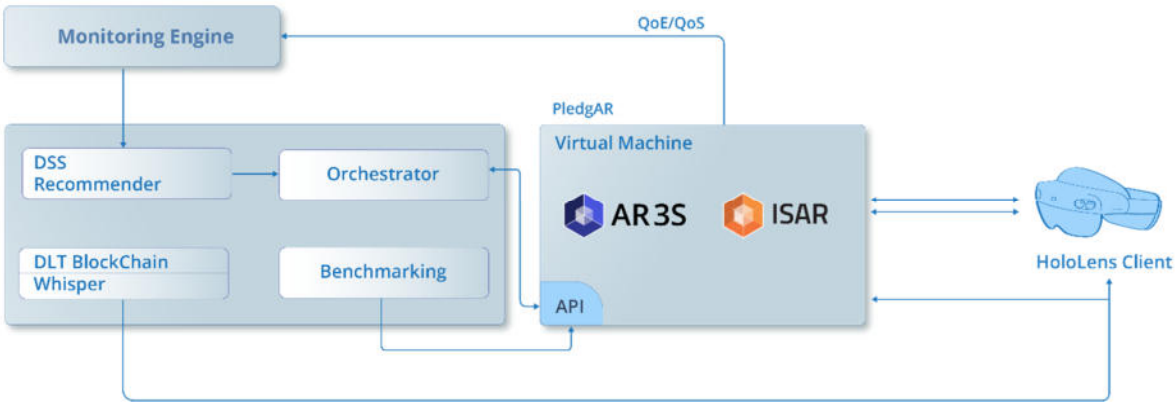

Figure 2: Schematic depicting the integration of the PledgAR Workspace with the Pledger toolkit

Orchestration and management of resources is provided by a connection of the PledgAR Workspace with the Pledger Orchestrator. As mentioned above, the PledgAR Workspace, which enables the remote rendering and streaming of AR CAD content, is a Unity application running on a Qemu KVM [14] virtual machine located on a Linux PC, representing the edge node in this UC. An application programming interface (API), using NodeJS web framework, called Express.js is used to control the VM through commands received from the Orchestrator. Orchestrator driven API control of the PledgAR Workspace enables the start, stop, and optional upgrade of the VM. The latter is a highly relevant action, particularly given that it is determined by feedback from metric monitoring. Relevant metrics, such as file load time and latency, are sent from the PledgAR Workspace to a monitoring engine. These metrics respectively refer to the measured loading time of CAD files and the temporal delay of user movement, relative to replicated movements in the virtual space, are sent from the PledgAR Workspace. Metric information is published to the Pledger Apache Kafka [15] topic of the StreamHandler platform, from which SLA Lite uses the messages to detect deviations from set service level agreement (SLA) standards. Integration into the StreamHandler by the Unity-based PledgAR Workspace is accomplished by Confluent Kafka library for the .NET framework [16]. SLA violations are then used by the decision support system (DSS) to determine and recommend reallocations and resource adjustments to the Orchestrator for subsequent instruction of the VM and PledgAR Workspace. This feedback mechanism, along with input from Benchmarking results, ensures a high degree of versatility in PledgAR Workspace-derived AR performance which is tailored to the working demands of job-tasks. As a result, a continual upkeep of QoE for users is maintained.

As mentioned above, a high degree of security is a necessary feature during industrial MR experiences. ISAR technology makes use of the Web Real-Time Communication (WebRTC) technology to establish peer-to-peer connection between end-user devices and the PledgAR Workspace. A signalling event (handshake) establishes this connection; however, this step would benefit from increased security measures, and as a result was the focus of the third integration site with PledgAR. Pledger's blockchain network, the distributed leger technology (DLT), was used to handle this need and provided a framework for the Whisper protocol which contributed to the secure double encryption. This protocol contributed

to the generation of highly secure signalling between the PledgAR Workspace and relevant end-user devices, therefore ensuring a substantial improvement of the degree of security during data transmission.

In summary, the integration sites of Pledger, which respectively consist of connections with the orchestrator, the DSS/Recommender, and the DLT components, provided the necessary theoretical and technical framework to dynamically enhance machine-human AR experiences for end-users and stakeholders.

Table 1 gives an update of the table presented in the deliverable D5.4 [2], showing the status of the different integration endpoints.

It shows the components involved, the responsible partners for achieving the integration, data types and protocol used, as well as information about publisher and subscribers (if applicable) and the status of integration at the time of writing this deliverable. An identifier is assigned in the schema X.Y.Z, where X is the number of the UC, Y is the core component (Orchestrator/E2CO – 1, DLT – 2, SaaS/IaaS – 3, SOE – 4, RAN – 5 and StreamHandler – 6), and whereas Z the UC component: VM Configuration – 1, Client Utility Application – 2.

Table 1: Status integration endpoints UC1

Integration Endpoint	Components	Responsible	Data Type & Protocol	Publisher/ Subscriber	Status M24	Status M30	Status M36
1.1.1	Orchestrator – VM Configuration	ATOS, HOLO	REST API, OpenVPN TCP	N/A	In progress	Done	Done
1.1.2	Orchestrator – Client Unity Application	ATOS, HOLO	REST API, OpenVPN TCP	N/A	In progress	In progress	Done
1.2.2	DLT – Client Unity Application	INNOV, HOLO	JSON, Whisper	N/A	In progress	In progress	Done
1.3.1	SaaS/IaaS Monitoring Engine – VM Configuration	ATOS, HOLO	JSON, Kafka via StreamHandler	Topic: ▸ UC-metrics	In progress	Done	Done
1.3.2	SaaS/IaaS Monitoring Engine – Client Unity Application	ATOS, HOLO	JSON, Kafka via StreamHandler	Topic: ▸ UC-metrics	In progress	Done	Done

4.2 Pilot deployment

Further details regarding the location of UC 1's pilot is described in the UC3 descriptions detailed below, but a general mention will be made here to provide context for the use case deployment of the PlegeAR Workspace. FILL has extensive experience in many industrial sectors and as such has the necessary on-site foundation and infrastructure to host the case scenarios of UC 1. FILL's industrial Future Zone therefore provided the necessary physical space and WiFi requirements for integrating AR applications. The latter is necessary for enabling the connections between PledgAR based ISAR streaming and integrations with Pledger framework tools.

As mentioned above, UC 1 is divided into three planned use case studies. These studies are mutual in their collaborative nature and consist respectively of multi-user interactions in virtual space during 1) fast prototyping, 2) remote service and support measures, and 3) during training events. In addition to needing a physical space and sufficient room for users to walk around in during these case studies, necessary hardware and software components were required. Although the required components have been listed above, they will be mentioned here with emphasis on their role during the pilot deployment. Following, general non-technical requirements will be stated which assisted in facilitating end-user safe and friendly deployment scenarios.

Given that this UC is intrinsically focused on interactions in, and with, augmented reality, the setup measures will be discussed with respect to the focus on AR interactions.

4.2.1 AR hardware/software setup requirements

Virtual content is viewed by end-users on the HL2 devices. These devices enable the viewing of holographic visualizations superimposed on the real-world space. Furthermore, the HL2 provides auditory input for collaborative communication. The battery life of the HL2 is roughly ~1 hour following constant use and therefore requires either prior charging or charging availability with a power bank via USB-C cable. With respect to the edge component, a laptop is provided (see above for specifications, including necessary charging upkeep materials) which provides the VM and PledgAR Workspace application. Specifically, within this Workspace, the ISAR SDK provides the streaming of virtual content to end-users and reciprocally returns audio and visual input. The connection between the HL2 and PledgAR VM is established through a Virtual Private Network (VPN) connection by OpenVPN. Edge-based ISAR streaming is beneficial in this effect as it enables the laptop to be at some distance from the user, which allows suitable space for interacting with virtual content. OpenVPN establishes an IP, and an open port is also required for this. A user interface (UI) on the HL2 allows end-users to load CAD files for subsequent interaction.

4.2.2 Pledger Toolkit Setup Requirements

PledgAR/VM connection to the Pledger core toolkit is facilitated through the aforementioned VPN which enables the subsequent integrations as described above (e.g., orchestrator to VM, DSS to orchestrator, etc.). These connections are essential in order to experimentally assess the ability of Pledger to offer resource scaling responses during AR scenarios. Beyond connectivity requirements, SLA configurations need to be provided and set up in order to establish possible SLA violations with which to assess PledgAR responses. The integration of Pledger components with the PledgAR Workspace and AR end-devices requires only the previously mentioned laptop PC and HL2, no other edge devices are needed.

4.2.3 Non-technical Setup Requirements

Ergonomic considerations for this UC 1 generated a few additional requirements for case study deployments. FILL's Future Zone, which is used for a physical space to deploy the UC 1 scenarios, provides sample space to move around and interact with the virtual objects, while moving and removing the risk of unintentional physical collisions while participating. This feature is beneficial during corona-virus times, as suitable distance can be maintained by UC participants. Further considerations to this effect are provided by supplying suitable hygienic measures for the HL2. A supply of disinfectant is provided to ensure proper hygienic conditions of the HL2, which comes in direct contact with UC participants. End-user data safety measures are also put into place during setup of this UC. Participants are asked to fill out and sign documentation which instructed and informed on the actions taken by the UC provider to ensure proper handling of relevant end-user data, such as e.g., auditory information picked up from the HL2, recorded visual feed flow, and user hand calibration metrics.

4.3 Experiments performed

The specific experiments performed in this UC are structured around the three case studies described above. Each study utilizes the PledgAR Workspace to facilitate collaborative visualizations, communication, and/or manipulations of holographic 3D content in different contexts. Across each study, the experiments presented here profiled application and end-user metrics which are published in real time and assess the quality of service (QoS) and quality of experience. The results of these experiments provided a qualitative and quantitative view of the impact and influence of Pledger during AR collaborative sessions. Accordingly, UC 1 intended to appropriately challenge ISAR virtual streaming to investigate the extent to which Pledger edge computing is able to assist in dynamic modulation of performance quality.

Each study was deployed by a local team at the FILL Future Zone site (described above and in detail in UC3 below) consisting of internal members of the consortium. Participants involved were equipped with the necessary end-user hardware as detailed above. The following sections detail the case study experiments and their specific aims.

4.3.1 Fast Prototyping

The first case study was intended to simulate fast prototyping events of vertical-specific industrial content. During this case study, transformation of a variety of file formats into three-dimensional holograms permitted highly interactive sessions of product presentations. Furthermore, manipulations of these three-dimensional files with multiple users facilitated a collaborative and agile approach to developing these products. Virtual prototyping is challenged by the need to provide seamless resolution and maintain a high quality of the AR experience despite the possibility of extremely large CAD files which are loaded in. The Pledger framework has offered a solution for this, in the ability to appropriately scale and allocate resources to handle the loading of such variable files. Here users loaded and subsequently virtually engaged with CAD files of different sizes and then each user was asked to verbally report on the extent to which hologram quality, and thus service quality, was maintained from their perspective during the session. Prior to these interactions, when utilizing the PledgAR Workspace, SLA violation thresholds were defined and configured. When provided with a particularly large CAD file, SLA violation execution and subsequent decision, recommendation, and orchestration of resource allocation occurred. The resulting efficacy of such responses was assessed by analysing latency and file loading times. These metrics were important in determining the functional success of these situations. This case study ultimately aimed to provide information on the ability of the Pledger framework to appropriately modulate performance metrics.

4.3.2 Remote Manufacturing and Service Support & Training Interactions

In the second and third case studies, a more targeted scenario was used. Rather than an open collaborative “research and development” experience, the session in this study aimed to provide specific assistance during a situational problem. Due to the similarity in multi-user environments between the two, they will be discussed in tandem.

In the former (case study 2), diagnostics of complex machinery, particular in the event that machinery is malfunctioning or broken, is the motivation of this case simulation. Users were provided with a live product (e.g., floor machine component) which was overlayed with a virtual copy of the product. Comparative analysis of these two products (i.e., virtual and real) enabled users to assess any differences. A high degree of resolution and appropriate overlaying is required in situations such as these, particularly given that industrial machinery can be quite complex and multifaceted. In addition, maintaining a consistent pixel stream without interruptions is required to ensure comparison can be made accurately. This case scenario assessed these features in a similar fashion as in case study 1, where the resolution and stream quality of the loaded file was assessed with focus on latency and load time metrics. Accordingly, users reported on the quality of the overlay and interaction with it.

In both case studies two (2 - Remote Manufacturing and Service support) and three (3 – Training procedures), multi-user interactions were a focus. Industrial usage of AR is suitable for collaborative efforts. Whether the collaboration is occurring as a result of technical support (i.e., in the case of assistance with defective product) or the result of job training, such conditions can place a significant computational load on 3D streaming services. The Pledger framework was assessed here for the extent to which it can assist in the accommodation of multi-user contributions to virtual sessions. Provided with either an overlayed product for inspection (2) or situational training (3), multiple users will be integrated into the PledgAR Workspace. Quality of service and experience was the target goal to investigate here, particularly in the event of multiple users interacting with virtual content. As mentioned above, relevant application metrics were be fed back into the Pledger platform and the resulting ability of resource allocation was measured.

4.4 Feedback on using Pledger

This section provides a summary of the responses to the developer questionnaire which aimed at providing feedback and final impressions on the Pledger platform and its integration with the UC 1 application.

As an overall summary, the usage of Pledger in the above application of this UC was viewed rather favourably by the development team. With respect to its intended functionalities, the edge computing contributions to AR streaming performance were considered to be beneficial. The most useful functionalities were reported to be in the establishment of the SLA violations and the integration with the DSS/Recommender components. Combined with the Orchestrator and its control over the VM/PledgAR Workspace, these specific features were able to directly satisfy the currently identified shortcomings in AR use in industrial settings (see above). A few constructive critiques were stated, particularly with respect to the added steps required for launching the VM and connecting to the remote service. However, the beneficial contributions of the Pledger system were noted to outweigh these minor inconveniences. The benefits of SLAs, particularly with respect to indexing against end-user device and application metrics, and subsequent modifications in AR streaming performance, were considered to be instrumental for ensuring QoE for end-users. For industrial AR technological solutions, such features are important and noted by the development team to be ideally executed with Pledger.

Application experiments were able to be executed with ease and operated efficiently without significant limitations. Dynamic resource reallocation was able to be correctly executed as per expected outcomes. In the establishment of these integrations, relevant documentation was indicated to be relatively easy to understand. In the event that difficulties in interpreting such documentation were encountered, the development team stated that support was extremely helpful, efficient, and functionally meaningful for the application. The resulting integration of Pledger components was considered to go smoothly most of the time; however, a couple of situations were unfortunately noted to have some minor difficulties (e.g., specifics of having to building the VM and also when using the DLT).

When compared to previously used technologies, the development team rated Pledger as a 3 on a scale of 5 and believes that the corresponding results here would not be able to be generated achieved without its use. Feedback on possible future implementations was considered. Some Pledger components currently support only containerized applications (e.g. AppProfiler) and developments to extend the functionalities to support VMs in the future are highly appreciated. In general, the integration and use of Pledger in this application was favourably viewed throughout the entire process, ranging from conceptualization, integration, and implementation.

5 Use Case 2: Edge infrastructure for enhancing safety of vulnerable road users

The widespread introduction of electric scooters and bicycles in cities has led to a significant growth of micromobility as a popular means of transportation. Even though in general the number of road accidents has consistently decreased across Europe, the number of accidents in cities tend to increase, as these new means of transportation are introduced. It has become clear that the increment of micromobility as means to commute comes with a series of challenges for the cities to assure the citizens' safety and to reduce potential accidents: new regulations to address the shared use of spaces used by pedestrians and electric scooters become necessary to assure harmony between these groups. New laws, such as the imposition of speed limits or mandatory assurances, need to be evaluated and implemented to control the market and protect both vulnerable road users (VRUs) that include pedestrians but also micromobility users and motorized vehicles. Further, infrastructures need to be adapted, e.g., with clear street layouts and dedicated lanes for VRUs to reduce potential risks and establish safe spaces in which VRUs can circulate.

Looking at Barcelona as a reference, in spite of the many changes introduced during the last years to improve the safety for VRUs and to assure a safe usage of public spaces for all groups (VRUs, vehicles), the existing street layout can often be complex and difficult to grasp, meaning that dangerous situations still might arise.

The UC2 developed in Pledger looks at ways to increase the safety for VRUs by introducing the use of dedicated edge infrastructure to deploy a service that can help to raise the awareness of VRUs about risky situations, with the final goal to reduce potential accidents. The Risk Detection and Notification Service (RDNS) deployed and managed by Pledger in a city-wide infrastructure is aware of electric scooters and the public tram in and around a tram station (Fluvià), alerting VRUs of a potential risk every time scooters approach the tram station while pedestrians are entering or exiting the tram. Due to the layout of the tram station, this situation incurs in possible collisions between scooter users and pedestrians that can result in serious accidents, considering the speed of scooters can reach up to 25 km/h.

The UC relies on gathering information about electric scooter users via the IEEE 802.11p protocol to collect vehicular data coming from on-board units (OBUs) of electric scooters that is shared with the RDNS running in the compute infrastructure via vehicular road-side units (RSUs) that allow OBUs to connect to the infrastructure. Further, the tram is also equipped with an OBU, so that its position is also shared with the RDNS. Based on the location information obtained from these devices over vehicular communications, it is possible to detect whenever tram and scooters are both present at a tram station and to alert the scooter users about potential collisions with pedestrians. For that purpose, a gadget that can be installed on the electric scooters has been developed that gives an audio-visual warning (buzzer + LEDs) to the scooter rider, indicating to reduce speed and be cautious of the surroundings, reducing the likelihood of potential accidents.

Another aspect relevant to this UC is the fact that Pledger is used to create a network slice across the city-wide infrastructure that reserves dedicated radio and compute resources for the RDNS. This is a prior step which is performed before the application is deployed, in order to ensure that the services will obtain the desired resources and do not starve on resources. In addition, the network slice in which the UC application is deployed (formed by radio access, network and compute resources) is isolated from other slices, thus guaranteeing that UC2 application can coexist in the city infrastructure.

5.1 Integration with Pledger

Apart from the development of the RDNS application that implements all the features to detect possible risks and to send out alerts, a series of integrations between the application and Pledger and also between different Pledger modules have been necessary to implement all required features for this UC.

5.1.1 SOE Framework – RAN Controller

The Slicing and Orchestration Framework and RAN controller core modules of Pledger, and on which UC2 relies on to create a dedicated slice with compute and RAN resources, required of a new integration to support the IEEE 802.11p technology featured in UC2. IEEE 802.11p has been selected as it is the standard technology for wireless access in vehicular environments. Cellular radio alternatives such as Cellular Vehicle-to-Everything (C-V2X) have been disregarded due to constraints related to the granting of permissions for deployment and allocation of dedicated spectrum by relevant public authorities. Prior to this integration, other radio technologies not available in the UC, like IEEE 802.11ac, 4G or 5G New Radio (NR) from certain vendors were already supported, meaning that the remote and centralized physical configuration of radio elements and the reservation and configuration for a slice and a particular service were implemented for those technologies. The support was missing for the IEEE 802.11p technology required for UC2, as such it had to be integrated first with the RAN controller (in its role as module that configures the radio devices) and then with the SOE Framework (in its role as module that manages the slices and requests, among other things, the physical and slice-related configuration of radio elements to the RAN controller).

The integration on the RAN controller side mainly required the implementation of necessary remote procedure calls on a radio node equipped with IEEE 802.11p radio transceivers to configure the physical parameters of the said transceivers. Further, a set of adaptations had to be made in the RAN controller to allow for the SDN-based configuration of the data plane of the IEEE 802.11p radio transceiver, so that it can be attached to the network slice that hosts the RDNS, enabling data transmission between the IEEE 802.11p transceiver and the vehicular stack (as detailed below).

The SOE Framework had to be extended to support infrastructure configuration and slice creation featuring the IEEE 802.11p technology. Apart from introducing this new type of radio interface to the list of supported technologies and available configuration parameters, another important aspect had to be considered: each IEEE 802.11p interface of an infrastructure used for vehicular communications needs to connect to the so-called Vehicle-to-Everything (V2X) stack, a piece of virtualized software developed by i2CAT that implements standardized vehicular message processing. In particular, the radio interface needs to connect to the V2X-communication (V2XCOM) module, a virtual service part of the V2X stack, that needs to be deployed when a slice with IEEE 802.11p elements is created. Note that all the V2X stack components, namely the V2XCOM and the Message Queuing Telemetry Transport (MQTT) broker, reside in i2CAT's infrastructure as Kubernetes pods, and these are automatically deployed when a V2X slice is created from PLEDGER's ConfService. The V2XCOM components receive Cooperative Awareness Message (CAM) messages containing localization information from the OBUs, and send out Decentralized Environmental Notification Messages (DENM) to alert OBUs of road hazards. The SOE needs to guarantee a strict mapping between each RSU that features an IEEE 802.11p radio and a dedicated V2XCOM module. This 1-to-1 mapping of a physical RSU and V2XCOM module is necessary to geographically distinguish vehicular information coming from different OBUs connected to the infrastructure and to be able to send information or warning to specific OBUs in the infrastructure. The logic and necessary steps to assure the layer 2 connections based on VLANs between the physical RSUs and the dedicated virtual V2XCOMs had to be implemented from scratch and required adaptations both on the SOE Framework and the RAN controller.

In addition, the SOE Framework had to be extended to support cloud-native infrastructures and applications, specifically those based on Kubernetes, which is an underlying technology in UC2. Prior to these developments, the SOE Framework supported infrastructures based on Openstack only. After the required implementations, the SOE Framework now has cloud-native slicing capabilities, and it

supports the deployment of network slices where compute chunks (one of the building blocks in Pledger's approach to network slicing) are based on Kubernetes namespaces. In this manner, the management of virtualized resources is delegated to Kubernetes, which implements a soft approach to resource sharing. The SOE Framework implements a set of additional resource sharing rules that are parsed to Kubernetes. Moreover, some additional implementations have been carried out to support the deployment of Helm charts in Kubernetes, through Open-Source Management and Orchestration (OSM); this feature is exploited upon the deployment of V2X slices, where the V2X stack is deployed in Kubernetes with the necessary configurations, depending on the infrastructure topology.

5.1.2 RDNS – DLT (storing data on the blockchain)

During application runtime, on-street users equipped with OBUs (VRUs, tram) connect to the RDNS application. User type information includes, e.g., whether the user is a bicycle/scooter or a public transport vehicle (tram), as well as the location of each OBU as collected by the RDNS application. This data, though being anonymous in the sense that it does not allow to identify particular persons, is considered sensible, as it reveals information about OBU locations and every time a possibly dangerous situation is detected. In order to safely store this data and make it accessible to trusted users, such as for example the city tram service or the mobility department of the city council, Pledger's DLT is used: periodic log dumps of tracked OBUs and registered events are dumped to the DLT through StreamHandler. Figure 3 shows an example log as sent to StreamHandler by the RDNS, where the position of a particular OBU and the tram is displayed, along with the alert type that is sent to the user.

Figure 3: Snippet from a log showing an OBU connected to the application (station_id 2).

For this purpose, an integration was necessary that involved determining a consolidated data format for the traces and logs, as well as creating a Kafka exporter that is capable of sending the RDNS application logs to StreamHandler on the topic "vru_positions". The logs are in turn consumed by the DLT through a subscription to the "vru_positions" topic.

5.1.3 RDNS – Monitoring Engine

In order to be able to implement SLAs for the RDNS application and to expose key metrics to the service provider that deploys the RDNS application, an integration between the RDNS and Pledger's Monitoring Engine was necessary. On the application side, the steps to be performed in this integration were to identify key metrics for the application of interest to the service provider, to implement the metrics in the RDNS application, to deploy a Prometheus [17] exporter in the RDNS application. On the infrastructure side, a Prometheus server had to be deployed in UC2's Kubernetes cluster, and a metrics scrapper had to be implemented to collect the metrics from the RDNS's Prometheus exporter. Finally, UC2's Prometheus server was integrated with Pledger's Monitoring Engine. The metrics can be retrieved by UC2's service and infrastructure providers on UC2's side and, in addition, these can be visualized on the RDSN's Grafana dashboard.

Moreover, on the Monitoring Engine's side, the integration with UC2's Prometheus server can be exploited to define rules and implement corresponding SLAs. Relevant examples include the service availability or end-to-end delay of the RDNS application, which is a metric generated by the application; and the service availability, which is based on a custom Prometheus query and which, together with the end-to-end delay, have been used to create an SLA that is fed to the DLT for the configuration of smart contracts rules and the update of the wallet balance.

5.1.4 RDNS – DLT (Smart Contracts)

Apart from using the DLT for storing the vehicular logs generated by the RDNS, it can also be used to create and apply smart contracts. These smart contracts are created on top of SLA agreements that are based on metrics collected from the application and infrastructure. In particular, for UC2, two SLAs are created, one for the service availability and another one for the end-to-end delay crossing the maximum threshold. The service availability is calculated as the average time over a twenty-four-hour period that the application is running in the Kubernetes cluster without interruption, and the corresponding SLA rule determines the maximum percentage of time the service is allowed to be unavailable. Regarding the end-to-end delay, a custom metric has been implemented on the SLA, which determines the percentage of time that a maximum end-to-end delay of 1.5 seconds can be surpassed over a 24-hour period.

To properly implement these SLAs, the service provider (RDNS application provider) must agree with the platform provider on different thresholds (percentages) and associated severity whenever a threshold is crossed. Every degree of severity results in a different refund policy, whenever the SLA is not fulfilled by the platform provider.

The service availability is calculated by applying a custom Prometheus metric that uses the availability metrics exposed by the UC2's Kubernetes cluster. The end-to-end delay is a metric calculated and exported by the RDNS application itself, and the associated SLA metric is also based on a custom Prometheus query as a function of the end-to-end-delay.

Summing up, UC2 requires a total of 4 specific integrations with the Pledger platform, the updated status is summarized in Table 2, showing the progress of the different integration endpoints and the status at the end of the project. The abbreviation follows the pattern explained in Section 4.1. The UC components are encoded in the scheme: Barcelona Infrastructure – 1 and Risk Detection and Notification System (RDNS) – 2.

Table 2: Status integration endpoints UC2

Integration Endpoint	Components	Responsible	Data Type & Protocol	Publisher/ Subscriber	Status M24	Status M30	Status M36
2.1.1	Orchestrator – Barcelona Infrastructure	ATOS, i2CAT	JSON, OpenVPN TCP	N/A	Done	Done	Done
2.2.2	DLT - RDNS	INNOV, i2CAT	JSON, Kafka via StreamHandler	Topic: vru_positions	In prog.	In prog.	Done
2.3.1	Saas/IaaS Monitoring Engine – Barcelona Infrastructure	ATOS, i2CAT	JSON, HTTP	N/A	Done	Done	Done
2.3.2	SaaS /IaaS Monitoring Engine - RDNS	ATOS, i2CAT	JSON, HTTP	N/A	In prog.	In prog.	Done
2.4.1	SOE – Barcelona Infrastructure	i2CAT	JSON, REST API	N/A	In prog.	Done	Done
2.5.1	RAN Controller– Barcelona Infrastructure	i2CAT	XML, NETCONF	N/A	In prog.	In prog.	Done

5.2 Pilot deployment

The UC2 pilot consists of a series of repetitions of the UC-scenario defined in the following subsection, involving internal and external users (i.e., users who are not involved in the Pledger project). For the UC2 pilot to be executed, the planned on-street deployment at the Fluvià tram station has been finalized during the last semester of the project. The Pledger deployment features custom-made radio nodes (single board computers equipped with IEEE 801.11p radios) mounted on lamp posts to act as RSUs, compact far-edge compute nodes collocated with the radio nodes, a rackable edge compute server deployed in a data center close to the lamp posts, and the necessary network infrastructure based on Ethernet and optical fiber to connect these elements with each other and with the main data center located in i2CAT's premises, on the other side of the city. Figure 4 shows a view on Barcelona city with its two locations that are around 8 km away from each other (connected over optical fiber). Note that both locations feature compute nodes: the main data center DC (which is split between i2CAT's headquarters and the adjacent Omega building), the edge server in ca l'Alier and the far-edge nodes collocated with the Wi-Fi nodes on-street.

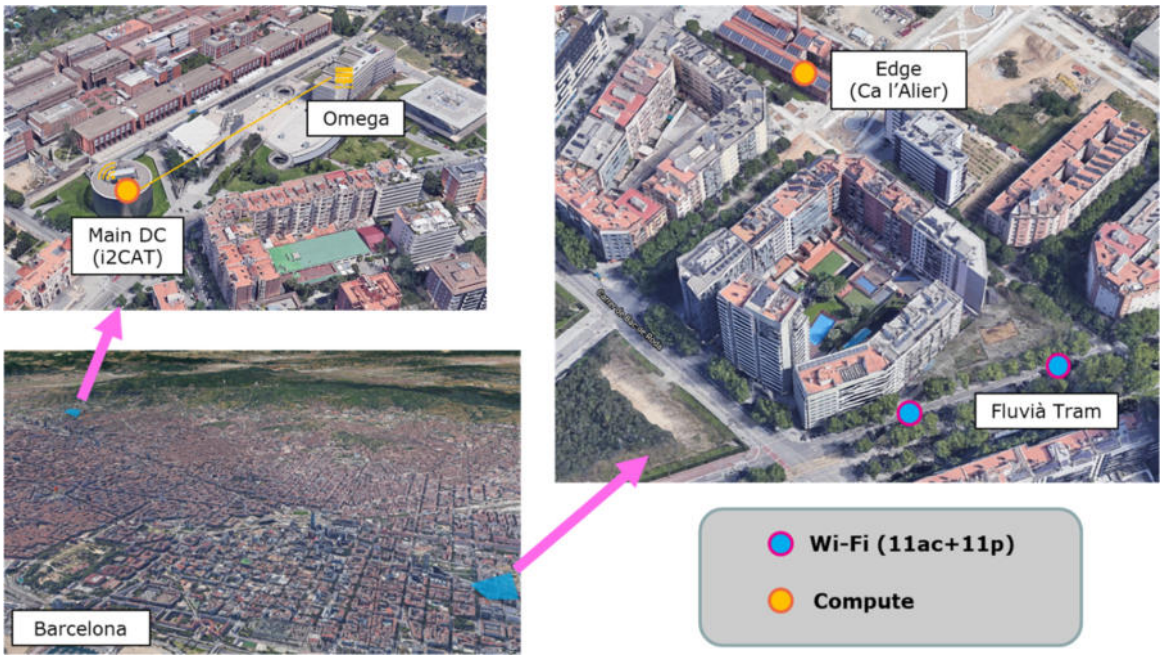

Figure 4: Pledger testbed in Barcelona with the main DC at the i2CAT premises in Zona Universitaria, as well as the edge and on-street deployment around the Fluvià tram station.

The Fluvià tram station has been chosen since it serves as a good showcase of a street layout that can lead to dangerous situations between scooter drivers and pedestrians. A risk exists, since the bicycle lane is located between the tram stop and the pedestrian lane, forcing pedestrians to cross the bicycle lane in order to access or leave the tram (see Figure 5).

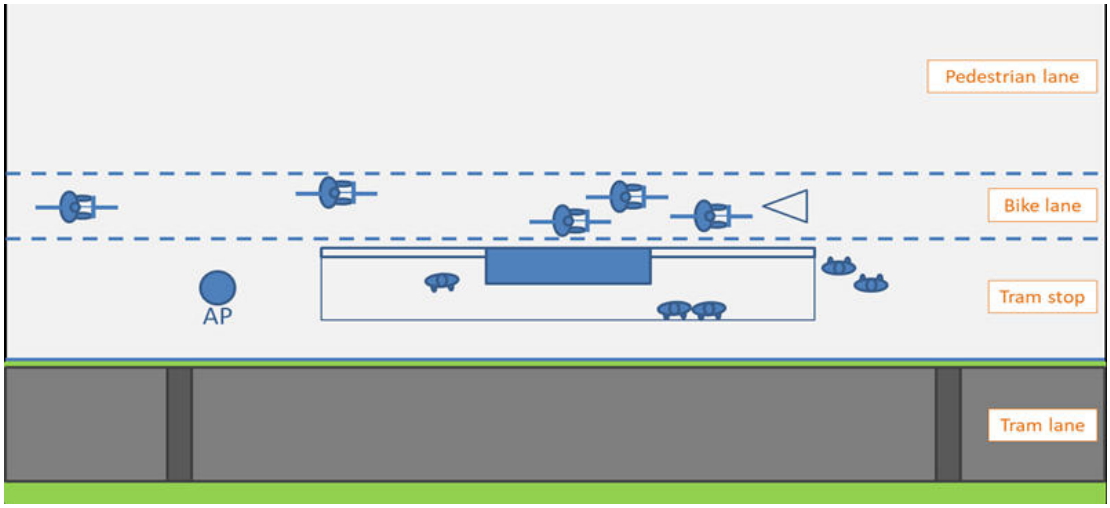

Figure 5: The layout of Fluvià tram station (and other stations in Barcelona), where the bicycle lane (middle), separates the tram stop area from the pedestrian area.

In real life tests at the tram station, we were able to observe that scooter or bicycle drivers often approach at high speeds, and pedestrians exiting the tram are distracted, e.g., listening to music or being busy with their phones, as they step onto the bicycle lanes and they do not pay attention. The scooter/bicycle drivers then are forced to perform (dangerous) evasive maneuvers or hit the break harshly. It becomes a matter of taking the right decisions in a few seconds or less, determining in glimpse of an instant whether an accident happens or not. The RDNS application intends to mitigate this risk, by sending out alerts to

scooter riders that can also be heard by pedestrians close by. In the following, the technical details of this solutions are explained.

The radio connectivity provided by the RSUs mounted on the lamp posts is necessary for the OBUs associated to scooter drivers and the tram to connect to the infrastructure and to interact with the RDNS application. The hardware and physical configuration of the IEEE 801.11p interfaces was designed in a way that, along the street in which the tram circulates, users can connect from up to 300 meters of distance. This allows for an early detection of scooter and also the tram, whenever they approach the tram station. The far-edge nodes are physically collocated with the radio nodes: connected over Ethernet via a switch, the radio node and the far-edge node both are both mounted on top of the lamp post. Since the Kubernetes cluster of the Barcelona infrastructure includes these far-edge nodes, it is the closest location to the on-street users in which a service can be deployed, ensuring minimal delays. In UC2, the far-edge nodes are used to deploy the V2XCOM modules that interact with the radio nodes, in order to minimize network latencies. The edge server located in Ca l'Alier, is physically very close (< 200m) to the two lamp posts, to which it connects over dedicated optical fiber. The edge server has more compute resources than the far-edge nodes. Finally, the main data center located at i2CAT offers compute resources to deploy services as well, while at the same time it is being used to host some of the Pledger platform components, in particular the SOE Framework and the RAN Controller.

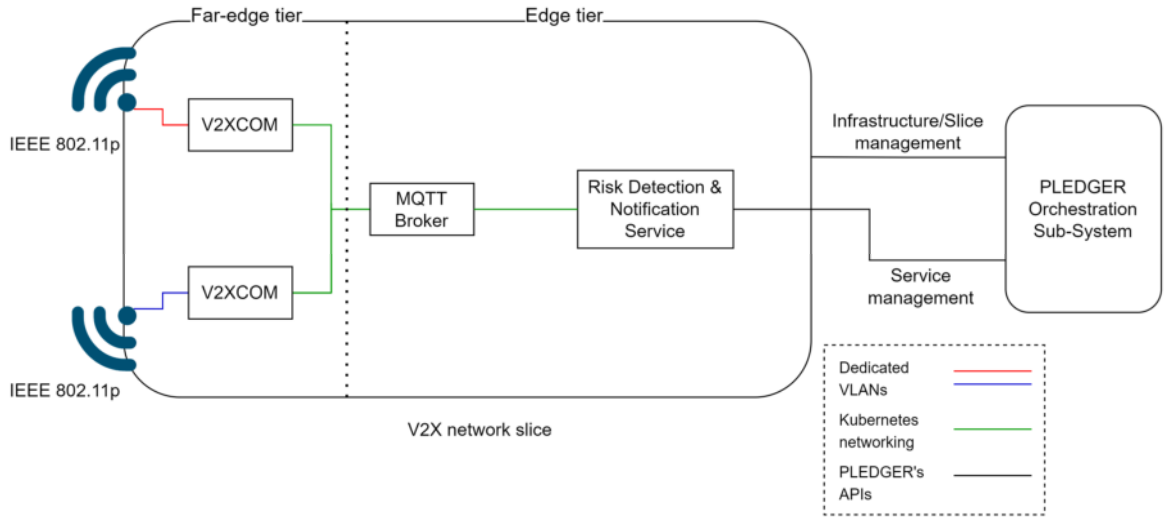

Figure 6: V2X Slice Diagram

With this infrastructure in place, Pledger can create a dedicated slice by reserving isolated computing, networking and RAN resources, and then deploying the RDNS application, enabling the risk detection features implemented by the application. Figure 6 represents the deployment of UC2 components over a dedicated network slice. It is shown that V2XCOM elements run on far-edge nodes, whereas the MQTT broker and the RDNS run on the edge tier. The PLEDGER Orchestration modules run on Engineering's cloud, except for the SOE and RAN Controller, which run on i2CAT's cloud tier.

5.3 Experiments performed

The pilot experiments performed in UC2 demonstrate the RDNS application and Pledger functionalities executed at the tram station Fluvià in Barcelona, where the Pledger on-street infrastructure is deployed, as described in Section 5.2. The pilot can be broken down into several steps.

The first step is to showcase that a dedicated slice can be created with Pledger on top of the city-wide infrastructure and that the RDNS application can be deployed. The slice creation is a basic step to enable

the physical radio connectivity on-street, to reserve the computing resources in order to satisfy the QoS of the RDNS application, and to guarantee isolation from other potential services that could be deployed in the infrastructure.

The second step is to showcase the RDNS application features: whenever a tram equipped with an OBU approaches or is stationed at the tram stop, any scooter or bicycle user that moves towards the tram station is alerted with an audio-visual warning emitted by a gadget installed in the vehicle. It indicates the risk of pedestrians possibly crossing the bicycle lane as they enter or exit the tram.

The third and last part of the experiment is all about showcasing relevant Pledger features: lifecycle management and placement decisions to assure the QoE of the RDNS application, safely storing logs to the DLT for trusted access, and smart contract creation for SLA agreements. The impact of these features on the system performance is evaluated partially on-street and partially remotely.

During the pilot, these 3 steps are repeated several times, for us to be able to observe and measure QoS and QoE aspects. All the experiments are performed internally (with Pledger members from i2CAT and IMI). In addition, experiments performed on-street are also demonstrated to external users, i.e., stakeholders that do not participate in the Pledger project.

5.3.1 Slice creation and service deployment

In the first step of the pilot experiments, Pledger is used to create a dedicated slice for UC2's application. Prior to this slice configuration, the infrastructure must have been configured by the infrastructure owner. The infrastructure configuration in this UC pilot is handled independently to the pilot execution, and it mainly determines which nodes will form part of the Kubernetes clusters and how the radio nodes are configured (operational frequency, transmission power). Via the ConfService, on top of this preconfigured infrastructure, a slice is created that includes a specific amount of computing resources across the far-edge, edge and main DC compute nodes, the two radio nodes deployed on-street, and the necessary networking configuration to enable end-to-end connectivity across these resources. Specific configurations for the radio, in this case the deployment of the V2X stack (since IEEE 802.1p is used), are also handled by Pledger in the slice creation phase. Once the slice is created, the Pledger orchestrator issues the deployment of the RDNS application. After this last step, the RDNS application is operational.

5.3.2 Risk detection and Notification System

The full functionality of UC2 application is demonstrated in this step of the pilot. The RDNS was already fully tested in the alternative indoor lab setup prior to the pilot, by injecting positioning[2] traces for both scooter and tram. During the pilot, however, the OBUs are installed in physical electric scooters driving around the tram station, and are also deployed in an actual tram, allowing for a real experiment with potential risks in the public space.

In the scenario performed on street, the tram is equipped with an OBU that connects to the Pledger infrastructure, so that the RDNS application can detect when it approaches. For the onboarding of the OBU, one of Pledger's team members from i2CAT enters the tram at the previous tram stop to Fluvià (Selva de Mar) carrying the OBU[3], exiting at the Fluvià station with the OBU. In parallel, one or two scooters equipped with OBUs and the alert gadget for audio-visual notifications are positioned at around 200 m of distance of the tram station, in the same direction from which the tram equipped with an OBU will be coming. When these two users see the tram approaching, they start driving towards the Fluvià tram station. The tram at some point overtakes them, leading to a situation where the tram is coming to a halt at the tram stop, while the scooter drivers are approaching the same tram stop. This will trigger the RDNS to send an alert to the scooters, urging them to slow down and be cautious, as they approach

[2] Based on Global Navigation Satellite Systems (GNSS) positioning messages.
[3] In the future, tram could allow the OBU to be deployed in the driver's cabin or to be integrated it with the tram vehicle. For the pilot this was not an option, since the OBU needs to be retrievable to repeat the scenario multiple times in a row.

the warning area defined in the application. Once the scooters arrive to the critical warning area, another warning will be used for imminent risk of collision. At this point the scooter drivers should have reduced their speed considerably and they should be on the lookout for pedestrians entering or exiting the tram.

5.3.3 Showcasing QoE assurance and DLT features

The final set of experiments performed during the pilot were related to showcasing several core features implemented by Pledger. On the one hand, these experiments demonstrate that Pledger allows to achieve the best QoE and on the other hand, they show that the DLT can be used for secure and trusted access to logs and traces and to establish smart contracts.

The set of experiments is comprised of the following:

Testing how the DSS suggests an upscaling of the application whenever certain behavior is observed in the metrics, indicating that the application is running low on resources. In practice, this upscaling is performed by increasing the resource limit quotas assigned to each pod on Kubernetes. We use simulated OBUs to increase the load imposed on the RDNS application and measure/observe the behavior of the application with the on-street users[4].

During the execution of the experiments as per Section 5.3.2, the traces and logs of the on-street users are collected and periodically exported to the DLT. We verify that this information is stored correctly and can only be accessed by trusted users. We use a combination of simulated and physical users (OBU on scooters).

We define two smart contracts related to thresholds for the service (RDNS) availability and the time the maximum allowed delay (1.5 s) is exceeded during operation. These two parameters are tracked based on the metrics gathered from the Barcelona Kubernetes cluster and the application metrics, respectively. For these experiments, we use simulated OBUs to create delay measurements.

5.4 Feedback on using Pledger

The results of the questionnaire filled by the i2CAT team members who participated in the integration of core modules and applications with Pledger yield very positive feedback on using and working with the Pledger platform.

In terms of transparency, developers appreciate the fact that Pledger handles the use of edge and cloud resources automatically, while optimizing the resource usage to assure QoS and QoE for the deployed services. Further, developers see a considerable benefit in being able to resolve tedious and time-consuming infrastructure configuration processes on the press of a button and the fact that dedicated slices can be created for different services on top of a common infrastructure. On the downside, before Pledger can properly manage a service, allowing for a smooth transition to the use of edge/cloud resources, the process of defining and configuring SLAs, metrics and triggers for the decision support system (DSS) takes quite some time.

However, developers are willing to accept this tedious process, considering the many useful features that are implemented by Pledger. Key features are slicing, easy configuration of edge/cloud infrastructures, and the mechanisms implemented by Pledger to ensure QoS requirements of applications are met. This is enabled by the DSS and the SLAs, which are highlighted by developers as key features. Further, being able to use the DLT for smart contracts is considered very useful.

Features that could be added to the platform include the support of maps for a visualization of an infrastructure, which would simplify and make the process of selecting computing and radio resources more intuitive and clearer, especially for larger infrastructures with several points of presence. It would also help to streamline the service deployment if all key functionalities, including the configuration of

[4] The simulated OBUs represent virtual users. These are necessary to create sufficient stress on the application that cannot be achieved with the two physical OBUs that have been constructed for the project.

SLAs and DSS algorithms could be done from one single screen in the dashboard, rather than being spread across different screens. Further, visualization in form of graphs that show the outcomes of benchmarking and app profiling processes for the infrastructure and service providers would be nice to have.

The services deployed with Pledger run smoothly and no issues are observed; however, at times, the support by developers of Pledger modules was needed to sort out minor issues. The services deployed vary between requiring a static number of resources, but can also be dynamic, i.e. the CPU and memory usage depend on the number of service consumers. Fortunately, Pledger considers services with both characteristics and ensures that the required resources can be provided to the applications[5] to meet QoS requirements.

In terms of productivity while using Pledger, the only limitation to be highlighted is the configuration of DSS algorithms and SLAs, as well as the deployment process of a service which could be simplified, taking several steps currently. Other limitations were associated to the scope of the UC2 service, in particular to the requirement of dedicated hardware for the service. This is especially important for the UC2 service that needs hardware to be deployed on street to provide radio coverage. Limitations coming from regulations of public usage, but also limited access to public space in general are highlighted by developers. No further limitations were observed in terms of productivity, which is thanks to the capacities of Pledger and also the documentation provided by the project. Developers consider it a good documentation when it comes to the guidelines of installing the platform components, however it might lack details to better understand how to use the platform. This has not been an issue for the developers though, given that support was always easily available from project partners responsible of the different Pledger core modules.

Similarly, the configuration, accessibility and the integration were also perceived as simple: very high ratings of 4.3/5 for the configuration, 4.5/5 for configuration and 5/5 for integration show that developers are satisfied with these three aspects. Overall, they rate the general usability of Pledger with a 4/5. It does not reach a perfect score, given the service deployment could be streamlined and the fine-tuning required to set up the desired behaviour of Pledger is not optimal. The same rating of 4/5 is also given to the level of improvement that can be obtained when comparing Pledger with the previous use of technologies. The fact that Pledger features cloud-native technologies is highlighted as a key improvement over previous use of VMs with OpenStack together with and the fact that slices managed by Pledger can support both technologies. This opens many possibilities on how infrastructures are managed: quick and easy deployment of slices with different types of virtualization and supported radio technologies. The addition of the DSS algorithms to ensure the QoS comes on top of this and is also highly appreciated, given that previously used solutions do not provide this and other related features. Developers agree with the fact that the automatization and features provided by Pledger were not available before and that they can make good use of them.

To improve Pledger further, they suggest additions to the interface/dashboards to expose more information, such as the status of a slice or the status of a services or functionalities that are currently active, as well as visualization of logs in case some configurations generate errors. While this information and these logs exist inside the different core modules, they are not exposed to a central dashboard/site. This goes in line with the previously mentioned streamlining of service deployment that is desired by developers.

[5] Always taking into account the physical limits of an infrastructure.

6 Use Case 3: Manufacturing the data mining on the edge

The UC3 supports manufacturing experts to determine the process stability, enabling the improvement of it in the area of metal processing of highest accuracy. Using the developed analytical services, the expert is able to analyse the process and find deviations in the process in three regards:

- Stability of media consumed by the machine (pneumatic and electric)
- Stability of parts produced by the machine in terms of cycle time and quality
- Thermal stability

The determination of the thermal stability is of high interest for the machine operator. The (metal) components of the machine are extending due to thermal influence during the machine movements until they reach a plateau. Unexpected downtimes or small operating delays have an influence on the cool-down of the machine, which can hardly be estimated by experts given the influencing factors (duration, shop-floor temperature, etc). The thermal stability of the machine is crucial, especially for processing with highest accuracy of 0.01mm. The duration of the warm-up process is configured by experience, which is usually longer than necessary to not risk any loss in quality of the parts, resulting in lower efficiency. Another module was developed to estimate a thermal stability performance indicator and the subsequent estimation about the thermal condition of the machine. The result is an instruction to the machine about the further proceedings (continue warm-up, go into production mode), which is transferred directly to the Programmatic Logical Control (PLC) on which the machine reacts within the time of one controller cycle. All developed modules are integrated into Cybernetics Analyze, which is FILL's Industrial Internet-of-Things solution for collection and processing of data digitizing the entire production process. All Cybernetics Analzye modules are deployed on the industrial PC, acting as an edge device. These PCs are very robust, with the drawback of limited resources.

Using Pledger, the limitations of an edge device regarding computational power can be overcome, by offloading applications to the cloud to ensure the QoS and associated SLAs of the applications. Furthermore, sensitive information obtained by the application, can be stored securely on the DLT giving access to authorized parties only.

6.1 Integration with Pledger

Pledger offers the following benefits to the UC, to enable the described advantages:

- Infrastructure management and service orchestration;
- Monitoring performance, QoS and SLAs;
- Infrastructure benchmarking and application profiling;
- Secure management of sensitive data;

This requires the establishment of connectivity and the integration with Monitoring Engine, DLT and StreamHandler for data transfer.

6.1.1 Connectivity

ConfService is the starting point to configure the infrastructure, as well as the UC applications and the associated SLAs to enable all subsequent operations. The connectivity to access ConfService as well as the UIs of the other Pledger components was established using an OpenVPN connection. Opening the Docker port of the edge node enables the Pledger components to access the infrastructure as well as the UC applications. This was described in detail in D5.3 [1]. This way, services can be orchestrated by the Orchestrator, Benchmarking Suite can create benchmarks of the infrastructure and applications can be profiled by the AppProfiler.

6.1.2 UC Application – Monitoring Engine

To establish the information flow in the other direction as well, and to provide application-specific metrics and metrics about the infrastructure usage, integration with the Monitoring Engine was established by pushing relevant metrics via StreamHandler.

These metrics are extracted in two ways. First, there is a microservice gathering in regular periods (every 10s) the actual usage of the different Docker containers running on the edge infrastructure in regular periods (every 10s). Furthermore, metrics from RabbitMQ and the consumption of messages are extracted. These metrics are gathered as JavaScript Object Notation (JSON) messages and are pushed to RabbitMQ. Another service is consuming these messages and publishing them on StreamHandler, from where they are transferred to the Monitoring Engine.

The application-specific metrics are extracted by the application itself and pushed to RabbitMQ, where the procedure is the same as for the Docker related metrics. All details are given in D5.4 [2].

An example of these metrics received on StreamHandler is given in Figure 7.

Topic	Key	Date	Partition	Offset	Headers
UC-metrics		2022-11-07	0	4532303	0
{"DocumentType":"SystemMonitoring","DocumentBody":{"meta_info":{"InfrastructureProvider":"FILL","InfrastructureName":"UC3-edge","NodeName":"nerve-host"},"metrics":{"cpu_stats":{"used_millicore":180.0,"total_millicore":3000},"memory_stats":{"total_megabyte":15840.559104,"used_megabyte":1997.0040121}},"Version":1,"MA":"ma-005620","TimeStamp":"2022-11-07T06:58:55.816573+00:00"}					
UC-metrics		2022-11-07	0	4532304	0
{"DocumentType":"RabbitMQMonitoring","DocumentBody":{"meta_info":{"InfrastructureProvider":"FILL","InfrastructureName":"UC3-edge","NodeName":"nerve-host"},"service":{"type":"RabbitMq","generated_at":"2022-11-07T06:58:56.518225","management_version":"3.8.4","queues":[{"name":"[LastAnalysis].[AirConsumption].[data.exchange.timestamp-recv]","type":"classic","state":"running","auto_delete":false,"consumers":0,"consumer_capacity":null,"consumer_utilisation":null,"durable":true,"exclusive":false,"message_traffic":{"total":0,"ready":0,"unacked":0},"source_exchanges":["data.exchange.timestamp-recv"]},{"name":"[Tier 1].[AirConsumption].[data.exchange.fast]","type":"classic","state":"running","auto_dele					
UC-metrics		2022-11-07	0	4532305	0
{"DocumentType":"DockerMonitoring","DocumentBody":{"meta_info":{"InfrastructureProvider":"FILL","InfrastructureName":"UC3-edge","NodeName":"nerve-host"},"service":{"container":[{"id":"bc82ea215dd81c1c3cb2c06ec9969492fd71a290961f721b3fe2ecea4993dd15","short_id":"bc82ea215d","name":"analyse_media_consumption_pneumatic","created":"2022-10-28T09:58:18.789243174Z","platform":"linux","image":"f0422be5dc018976e26cee6b82bd5244aed93f52500e6b9d6725a4b19e29bc29","image_id":"sha256:f0422be5dc018976e26cee6b82bd5244aed93f52500e6b9d6725a4b19e29bc29","status":"running","memory_stats":{"usage_megabyte":61.0304,"limit_megabyte":15843.71712},"cpu_stats":{"usage_millicore":1.3069805921052633,"limit_millicor					

Figure 7: Example of UC3 metrics on StreamHandler

6.1.3 UC Application - DLT

Even though no personal data is processed in UC3, some information about parts produced by the machine are considered sensitive. This includes the information about the number of parts produced in a given time range, as well as the number of rejects in this time range. This information reveals sensitive details about the utilization of the machine and therefore also about the shop floor. Using the information about the amount of parts produced and the number of rejects gives information about the running business (many parts and low rejects indicate a well-running business, whereas a low number of parts with high number of rejects indicates the opposite). Therefore, the information about parts produced are stored on the Pledger DLT to give access to this information only to authorized parties.

This integration endpoint connects the UC application with the DLT. This connection was established by pushing the relevant information of the application “parts-produced” as JSON to StreamHandler and storing the messages on the DLT. The procedure was implemented as described in D5.4 [2].

6.1.4 UC Application - StreamHandler

The UC3 offloads an application in case specific SLAs are violated and the performance of the application is reduced. As all analytical services depend on consuming machine data, which are processed via a message broker, they require a connection to a suitable message broker as a data source, regardless of their location of deployment. In case computation is offloaded, StreamHandler (which is based on Apache Kafka) is used as data source in any other location than the UC3 edge infrastructure. A component (Broker-To-StreamHandler) was developed, to publish the machine data on a dedicated topic on StreamHandler, where the application consumes the messages in case it is offloaded. To handle the different types of message brokers, the application was adapted, and the interface was standardized to handle both RabbitMQ and Kafka as underlying message broker. All details of this integration endpoint are given in D5.3 [1].

6.1.5 Smart contracts in UC3

The Pledger Blockchain can also be used to create and apply smart contracts. Like UC2, two SLAs are created. One for the service availability on the edge and another one for the calculation time of the thermal stability indicator. The service availability is calculated as the number of messages ready on the RabbitMQ (meaning, they are not consumed as the service is not available) multiplied by the frequency of the messages. For the consistency of result availability, the calculation time is used and a custom metric is implemented, evaluating the standard variance over a period of 1 min, which should not exceed a specified threshold. The threshold is determined and specified during the pilot operation.

To properly implement the service availability SLA, the service provider (thermal-stability application provider) must agree with the platform provider on different thresholds (percentages) and associated severity whenever a threshold is crossed. Every degree of severity results in a different refund policy, whenever the SLA is not fulfilled by the platform provider. The consistency of result availability SLA is also agreed between platform provider and service provider, even though the QoE for the customer is impeded in case the SLA is violated and could also be included in the refunding process at a later stage.

Table 3 gives an update of the table presented in the deliverable D5.4 [2], showing the progress of the different integration endpoints and the status at the end of the project, following the pattern as described in Section 4.1. The UC applications for the identifier are Edge Server – 1 and Basic Analytics – 2.

Table 3: Status integration endpoints UC3

Integration Endpoint	Components	Responsible	Data Type, Protocol	Publisher, Subscriber	Status M24	Status M30	Status M36
3.1.1	Orchestrator – Edge Server	ATOS, FILL	REST API, OpenVPN TCP	N/A	Done	Done	Done
3.2.2	DLT – Basic Analytics	INNOV, FILL	JSON, Kafka via StreamHandler	Topic: ▸ uc3-dlt	Not started	In progress	Done
3.3.1	Saas/Iaas Monitoring Engine – Edge Server	ATOS, FILL	JSON, Kafka via StreamHandler	Topic: ▸ UC-metrics	In progress	Done	Done
3.3.2	Saas/Iaas Monitoring	ATOS, FILL	JSON, Kafka via StreamHandler	Topic: ▸ UC-metrics	In progress	Done	Done

Integration Endpoint	Components	Responsible	Data Type, Protocol	Publisher, Subscriber	Status M24	Status M30	Status M36
	Engine – Basic Analytics						
3.6.2	StreamHandler – Basic Analytics	INTRA, FILL	JSON, Kafka via StreamHandler	Topics: ▸ fill-data-fast ▸ fill-data-result ▸ fill-data-timestamp	Done	Done	Done

6.2 Pilot deployment

The pilot is deployed in the FILL Future Zone, where a small manufacturing line is set-up for the duration of the pilot. The machine SYNCROMILL is set-up, and an industrial PC is installed in the cabinet control of the machine, acting as the edge node for UC3. The media-consumption services as well as the thermal-stability component are deployed on the edge node. Furthermore, Cybernetics-related services are deployed, i.e., backend, databases, and the message broker as well as all services related to the integration with Pledger to enable the communication with StreamHandler. Connectivity is established using OpenVPN as described in D5.3 [1]. The Cybernetics UI is made accessible to the end users in the shopfloor to enable the monitoring of the process and the determination of the process stability.

Before the actual start of the manufacturing process, the machine needs to be warmed-up to ensure the demanded quality of the parts produced. This offers the possibility to start with the first experiments.

The data about parts produced during the pilot are stored on the DLT and can only be accessed via authorized access.

Figure 8: Pilot machine in the Fill Future Zone

6.3 Experiments performed

During the pilot execution, several experiments are performed. As the operation is related to high effort and bound to some conditions and requirements in the shop floor, the pilot is executed in one run on 5 days to execute the different scenarios.

The relevant stakeholders are present at the same time during the pilot execution and are monitoring the process individually live and observing the analytical results on portable devices (laptop, tablet).

The experiments and the pilot execution can be broken down to 2 main steps:

- Warm-up of the machine, production and monitoring the process stability;
- Showcasing the Pledger features: lifecycle management and placement decisions to assure the QoE of the manufacturing process, safely storing logs to the DLT for trusted access, and smart contract creation for SLA agreements.

6.3.1 Service deployment and monitoring performance

Before the actual start of the machine and the production, all services to aid determining the process stability are deployed. Before this step, the infrastructure needs to be configured in the ConfService, which has been done already for the previous iterations. Using ConfService the applications are deployed: media-consumption, parts-produced and thermal-stability.

All applications are deployed on the edge, which is an industrial PC installed in the control cabinet of the SYNCROMILL machine. The computational power will be extended with cloud resources, especially, if resources get limited during the warm-up procedure.

6.3.2 Warm-up of the machine and manufacturing

The pilot experiments start with the warm-up of the machine to ensure the quality of the parts produced. Before the actual start, one part is produced with the cold machine and measured by a quality-check expert, to have a reference to compare with other parts. The machine runs a dry warm-up cycle moving the different axes with some tools and performing specified movements to ensure the extension of the metal components in a controlled manner. After the successful warm-up and as soon as the thermal-stability indicator gets stable, more parts are produced and measured.

The first experiments are performed only to monitor and evaluate the warm-up of the machine and to determine the benefits of using the thermal-stability component compared to traditional warm-up with predefined duration. Figure 9 shows the evolvement of the thermal stability indicator during this experiment.

After successful warm-up of the machine the production mode is turned on. Parts are produced and the cycle time and the stability of the process is monitored. The production is influenced by varying the speed of the machine, inducing downtimes in front of the production line, etc., and the process is monitored and the influence on the process stability is noted. Furthermore, the media consumption of the machine is monitored, and subsequently its stability is observed.

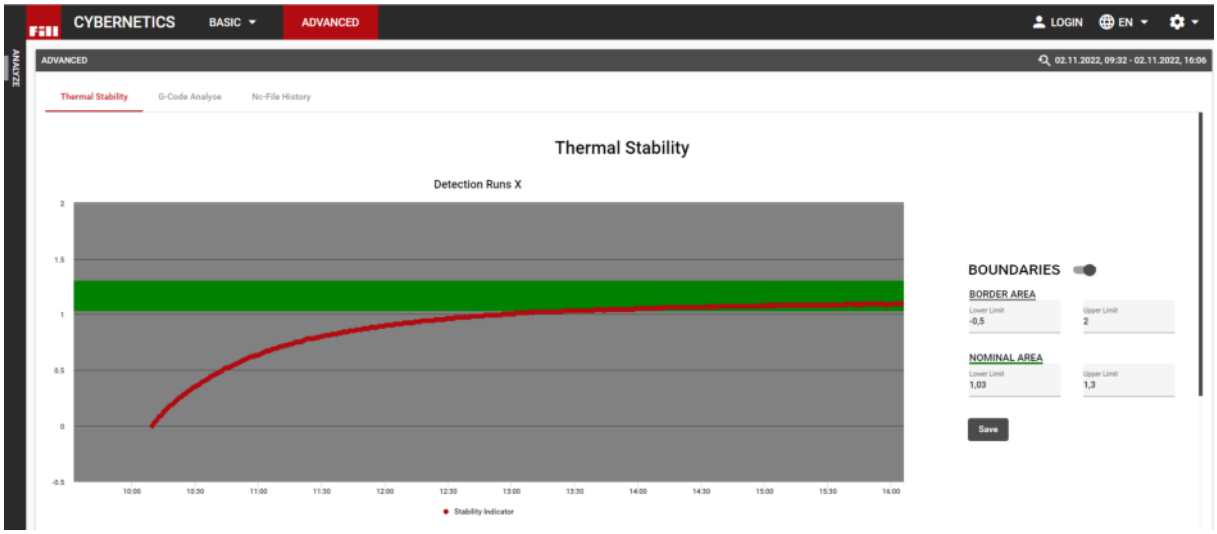

Figure 9: Visualization of the thermal stability indicator evolvement

6.3.3 Showcasing QoE, SLAs and improvement of performance

The UC has varying load on the edge – the thermal-stability component is not always under load, only during the warm-up phase of the production and after unexpected downtimes, when it is not clear, whether the machine is warm enough to ensure the desired quality of the parts.

In case of these phases, where the thermal stability needs to be determined, highest priority is given to this component to ensure the smooth process and to not lose any valuable time in the production.

The QoE is highly influenced by the calculation time of the thermal-stability component. In case the results are within a given standard deviation and within in a threshold, the warm-up process is perceived smooth by machine operators and no further effects on the mechanics due to uninfluenceable thermal effects due to cool down can be expected.

The calculation time required to calculate the thermal-stability indicator is determined, to establish a proper SLA threshold to ensure a stable and smooth warm-up process. This threshold is defined as an upper threshold of 0.09s, as the cycle time of the Programmable Logical Controller (PLC) is 0.03s and experiments showed, that three times the PLC cycle time is still conceived as acceptable. In case this

threshold is violated, the media-consumption app is offloaded to the cloud to free resources to the container. This is monitored in the Monitoring Engine as well as on the shop floor.

Using the information provided by Pledger, the QoE and the actual savings for the warm-up process are determined.

6.3.4 Accessing sensitive information on the DLT

The information generated by the parts-produced component, is stored on the edge, where only the end users have access to the it via the Cybernetics UI deployed on the edge.

The information about the number of parts as well as the number of rejects is considered sensitive, and stakeholders see the transfer of this information to the cloud for further use with critical eyes. However, this information opens the potential for new business models and innovation, e.g., use of digital services or sale contracts for machines based on pay-per-use.

During the pilot execution, the information about parts is stored on the Pledger DLT to guarantee the access to the sensible information (number of parts produced and number of rejects) only to authorized parties, e.g. customers/machine operators and Fill (machine builder). Surveys with internal sale personnel and product managers are held to estimate the increase in customers' trust and the potential for new business models derived from this technology.

6.4 Feedback on using Pledger

The use of Pledger and its components in UC3 can be categorized as follows:

- Configuration of infrastructure;
- Configuration of services and SLAs;
- Establishing connectivity;
- Integration with Pledger components to provide UC-specific information.

These steps were performed especially in the set-up phase of the pilot, whereas the functionalities of the system were explored after the application development and during the integration phases. The full benefits of the system were explored during the pilot operation phase.

During the development of UC3, the transfer of data to the cloud, as well as the optimal deployment infrastructure were explored. Configuring the service constraints and deployment options eases the transition of edge to cloud and vice-versa as the possibilities are transparent listed on ConfService and the actual deployment is centralized within the orchestrator without any further need to adapt the application. Developers found this concept very transparent.

The most useful features especially regarding application development are the AppProfiler, the Benchmarking and the Monitoring Engine, enabling the understanding of the nature for both the application and the available infrastructure. Especially for the thermal-stability component this was a very enlightening experience, as developers didn't have any experience with similar algorithms implemented in the UC. The measurement of the actual calculation time for the thermal-stability indicator revealed smaller numbers than expected. Experiments in the actual environment and considering the actual load on the infrastructure resulted in very valuable knowledge. They also confirmed that the offloading scenario is indeed beneficial to improve the quality of experience. The work on metrics and SLAs achieved in this project was the most valuable and would not have been possible without Pledger. This work will also be used and will be continued, to improve the product in the post-project phase.

A further functionality, which would be nice-to-have, is a notification system, where recipients can subscribe to be notified in case of SLA violations. This would be helpful for metrics and SLAs, where automated actions cannot help, and a manual intervention is necessary. This can be the case for connectivity problems (e.g., connection to the controller of the machine is lost) and other.

All experiments could be performed correctly and work as intended and expected. The resources required were dynamic, in the sense that different operations on the shop floor also use different load of the analytical services which are involved to determine the process stability during the execution of the operational work.

The documentation provided on GitLab is sufficient. Furthermore, a user-guide to especially provide some documentation on how to configure the different parts is currently under progress and will help with the replicability. The support of the technical partners was helpful, and they did their best to resolve any technical issues in a reasonable amount of time.

In UC3, integration with the Pledger system was mainly establishing over the connectivity to ENG K8s cluster, where the different components were deployed and opening the Docker endpoint on the UC3 infrastructure to enable the communication of the components with the UC3 applications and infrastructure. To enable the data transfer to the cloud, data was transferred via StreamHandler using dedicated data-topics. Via the authorization process, only authorized parties can consume messages from these topics. To provide UC-specific information to the system, i.e., metrics related to the performance of the UC, as well as metrics about the resources used on the infrastructure, integration with the Monitoring Engine was established via StreamHandler. Having StreamHandler as a central point for integration is one possibility to integrate with the system, there are also other options. However, as developers in UC3 are familiar with the concepts of microservices and publish/subscribe mechanisms, this concept was very intuitive to them, giving the opportunity to extend already existing systems to establish the connectivity with StreamHandler.

7 Multi Mobile Network Operator PoC

In addition to the 3 use cases defined in D2.1 [7], a fourth use case or rather a proof of concept (PoC) has been designed to showcase how Pledger supports the concurrent deployment of private 5G standalone (SA) networks for multiple mobile network operators (MNOs). In this PoC, two mobile network operators deploy each a 5G SA network on top of a shared infrastructure, using the Pledger indoor lab in Barcelona, along with 5G radio hardware borrowed from another project. Each private 5G network deployed in this PoC consists of: i) a dedicated 5G NR cell ii) a dedicated 5G Core, deployed over a dedicated compute chunk, iii) networking connectivity between the radio and the core. It is shown that this deployment can be done for 2 MNOs over the same infrastructure, but with dedicated resources.

Figure 10 shows the intended design for the 5G PoC.

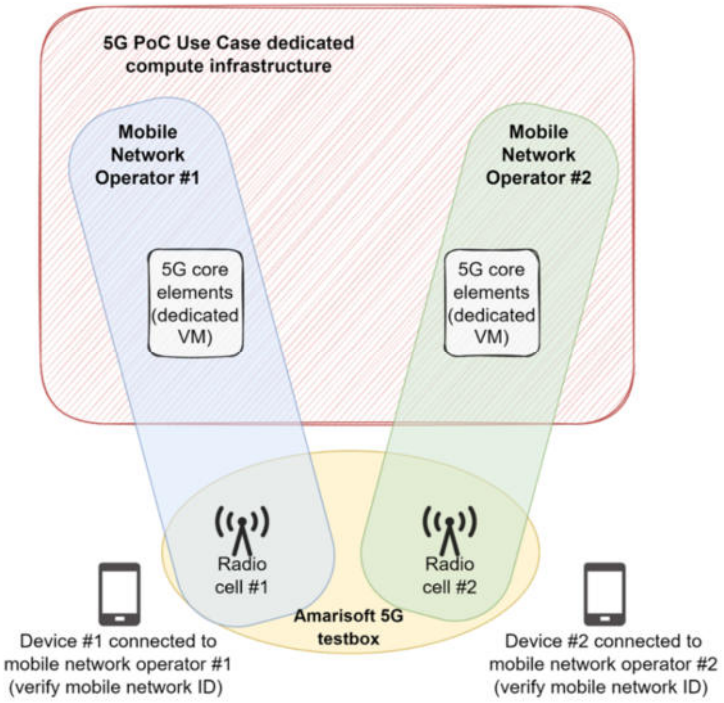

Figure 10: Design of the proof of concept to validate the 5G integration with Pledger.

The deployment and concurrent use of the radio and computing infrastructure by each of the MNOs, while maintaining isolation between the two 5G networks, is enabled with Pledger: a dedicated slice is created for each MNO that includes a 5G NR radio cell and the 5G core. Each slice allows users to connect to the 5G network and the two slices are isolated from each other, meaning that users active in one slice cannot access the other slice.

For this PoC, dedicated 5G radio hardware, namely an Amarisoft testbox, is used. It is a software-defined radio solution that enables the creation of several 5G NR cells with dedicated physical radio resources. It also allows to create a single cell that supports multiple MNOs via Multi-Operator Core Networks (MOCN)[6], although this functionality is not demonstrated. For the scenario showcased in this PoC, the total physical radio resources are split among the two MNOs, as each MNO requests a dedicated cell on different frequencies. The frequencies used for the two radio cells are both in the n77 band, for which the Spanish ministry grants permissions to use spectrum for research and experimental purposes.

[6] MOCN is an alternative RAN configuration also supported by the SOE Framework and RAN Controller.

Apart from a 5G NR cell, each MNO also needs a 5G core to be deployed as part of its private 5G SA network. For the PoC, we use virtualised Open5GS cores that are deployed via OpenStack: for each MNO (i.e., for each network slice), an Openstack project with dedicated computational resources is created (note, however, that there is a unique underlying physical infrastructure which is shared between both MNOs); then, for each MNO, an Open5GS[7] core is deployed as a virtual machine in the corresponding Openstack project. The way the association of radio cell with a 5G core works is as follows: During the slice deployment, if a 5G NR cell is selected for radio access connectivity, also an image of an Open5GS core is also deployed in the same slice. When bringing the 5G NR cell together with the dedicated 5G core over the slice- specific networking configuration, the cell starts radiating and allows users to connect. After this has been done for both MNOs, two distinct cells are radiating, each announcing a dedicated public land mobile network (PLMN) ID. User equipment (UE) can then connect to a cell, as long as they are registered with the MNO.

Further, a 5G NR metric exporter has been implemented for the Amarisoft hardware, that extracts key radio information, such as connected users per cell, uplink and downlink traffic per cell, or link quality information. In order for these metrics to be usable by Pledger (which can be exploited e.g., by the MNOs to check on the usage of the radio cells, or to define SLAs on top of these metrics to trigger warnings or notifications upon a breach), the metrics are gathered by a Prometheus instance in the RAN Controller module that manages the 5G NR radios. Just like other Prometheus instances used in Pledger, the Monitoring Engine can access the radio metrics and use them for various means, as mentioned above.

Thanks to Pledger, the entire 5G SA network deployment is fully automatized and happens upon the press of a button. Note that in this PoC no application/service is deployed along with the 5G SA networks, meaning that they just allow for basic user connectivity towards the 5G core and the Internet.

7.1 Integration with Pledger

For the 5G PoC, several integration activities had to be performed. 5G NR is a technology that was already supported by the SOE Framework and RAN Controller, the two core modules responsible for slice management and creation. As such, an integration with the Orchestrator module was required, in order to be able to push for the creation of the two slices from the "very top" of the Pledger architecture. When a 5G slice creation request is issued from the ConfService, the Orchestrator then executes a flow of API calls that instruct the SOE Framework to deploy and configure all the elements required by the slice; during some of these calls, the SOE Framework, in turn, instructs the RAN Controller to perform a series of actions necessary to provide radio connectivity.

The deployment of 5G slices is only supported for OpenStack-based infrastructures due to the Open5GS cores being virtualized as virtual machines, thus showcasing that Pledger supports different types of virtual infrastructure managers.

To be able to export radio metrics to the Pledger Monitoring Engine, the metrics exporter had to be deployed along with the 5G radio hardware and it had to be configured to export all the gathered data to the Prometheus instance dedicated to collecting radio metrics.

7.2 Experiments performed

In the first step of this PoC, the deployment of the two 5G networks is issued via the ConfService. Upon deployment, the Orchestrator communicates with the SOE Framework that processes the requests to create the slice, to request the radio configuration from the RAN Controller, and to deploy the 5G Cores, one for each slice. Once this is done, the two private 5G networks start radiating.

In the next step of the experiment, we use two UEs with different SIM cards, each one working with one particular MNO. The international mobile subscriber identity (IMSI) of each of the SIM cards is

[7] https://open5gs.org/

registered in one of the cores, allowing for the UE to register with one of the 5G NR networks. Once connected, the user can reach the Internet and it can ping the 5G core.

The final part of the PoC is about showcasing that the two 5G networks are isolated from each other: a UE connected to one of the 5G networks will not be able to reach the core (or any other services) that are deployed within the other 5G network. Further, we validate that radio metrics are correctly gathered in the Prometheus instance of the RAN Controller, demonstrating the connectivity of the two users connected to the two 5G NR cells deployed in the infrastructure.

8 Conclusions

In this deliverable, the work performed in the last iteration of the task T5.3 “Pilots operation and monitoring” was described. All UCs demonstrated the pilot deployment and execution in a relevant environment, mainly related to service orchestration, performance optimization and management of sensitive data. Furthermore, the use of the developed components in Task T5.1 “Application development for the Pledger use cases” in the pilots were demonstrated.

End users were involved during the pilots and feedback was gathered from them as well as from the internal developers involved in the project, to evaluate the benefits of using and integrating Pledger functionalities. All scenarios already outlined in previous iterations were successfully demonstrated end-to-end.

UC1 demonstrated the use of AR in an industrial setting and optimizing the performance based on the loading time and resolution of the AR stream. Pledger features used include the service orchestration, dynamic scaling of computational resources based on performance metrics and the management of sensitive authorization information.

UC2 implemented the pilot execution on street in Barcelona to reduce risky situations in public space. Pledger advanced the UC by reservation of dedicated computational and network resources, service orchestration and dynamic increase in computational resources, to ensure the notification of risky situations on time. Furthermore, sensitive logs were stored safely with authorized access.

Lastly, the pilot of UC3 was demonstrated in the manufacturing environment to ensure that no unexpected interruptions in the procedures due to delays in the analytical services are taking place. The UC benefitted from service orchestration, automatized offloading of computational resources, as well as management of sensitive information produced during the production.

Lastly, the Multi Mobile Network Operator PoC was presented. Even though, no Pledger UC directly implements the functionalities of 5G, Pledger can provide and manage 5G network resources. This was successfully demonstrated and described in this deliverable.

All UCs concluded the valuable use and advantages of using Pledger functionalities. Especially the performance monitoring and automatization of tasks (service orchestration, reservation of dedicated computational and network resources) were highly valued and also benefitted the development of the UC applications. Integration with Pledger components was conceived easy and transparent with excellent rating. The feedback was throughout positive overall, but also shows opportunities for future work and improvements, especially with respect to improving workflows and visualizations. However, this doesn’t impair the benefits UCs could achieve by using the platform. The results and observations achieved in this task are quantified and documented in the deliverable D5.8 “Pledger overall validation and evaluation” [8].

9 References

[1] V. Stanzl, "D5.3 Pilots operation and monitoring I", [Online]. Available: http://pledger-project.eu/content/deliverables. Accessed 20 October 2022.

[2] V. Stanzl, "D5.4 Pilots operation and monitoring II", [Online]. Available: http://pledger-project.eu/content/deliverables. Accessed 20 October 2022.

[3] I. Sarris, "D5.2 Pledger integrated demonstrator I", [Online]. Available: http://www.Pledger-project.eu/content/deliverables. Accessed 14 November 2022.

[4] P. Matzakos, "D5.6 Pledger integrated demonstrator II", [Online]. Available: http://www.Pledger-project.eu/content/deliverables. Accessed 14 November 2022.

[5] A. Betzler, "D5.1 Pledger Applications for the use cases I", [Online]. Available: http://www.Pledger-project.eu/content/deliverables. Accessed 7 November 2022.

[6] A. Betzler, "D5.5 Pledger Applications for the use cases II", [Online]. Available: http://www.Pledger-project.eu/content/deliverables. Accessed 7 November 2022.

[7] A. Betzler, "D2.1 Pledger Detailed Use Cases", [Online]. Available: http://pledger-project.eu/content/deliverables. Accessed 7 November 2022.

[8] V. Stanzl, "D5.8 Pledger overall validation and evaluation", [Online]. Available: http://pledger-project.eu/content/deliverables. Accessed 7 November 2022.

[9] O. Voutyras, "D2.3 Pledger Overall Architecture", [Online]. Available: : http://www.Pledger-project.eu/content/deliverables. Accessed 14 November 2022.

[10] "OpenVPN", [Online]. Available: https://www.openvpn.net/. Accessed 9 November 2022.

[11] "Kubernetes", [Online]. Available: https://kubernetes.io/. Accessed 14 November 2022.

[12] "ISAR SDK home page", [Online]. Available: https://holo-light.com/products/isar-sdk/. Accessed 03 November 2022.

[13] "HoloLens 2 home page", [Online]. Available: https://www.microsoft.com/en-US/hololens/hardware. Accessed 04 November 2022.

[14] "QemuKVM home page", [Online]. Available: https://wiki.qemu.org/Features/KVM. Accessed 04 November 2022.

[15] "Apache Kafka home page", [Online]. Available: https://kafka.apache.org/. Accessed 04 November 2022.

[16] "Confluent Kafka home page", [Online]. Available: https://docs.confluent.io/platform/current/clients/confluent-kafkadotnet/_site/api/Confluent.Kafka.html. Accessed 04 November 2022.

[17] "Prometheus", [Online]. Available: https://prometheus.io/docs/introduction/overview/. Accessed 23 November 2022

Annexes

Annex A

To gather feedback about using Pledger, its functionalities and integration, surveys were distributed among internal developers. The questions are divided into following categories:

- Transparency, concepts and functionalities
- Operation and execution of experiments
- Support and collaboration
- Usability of the system
- Results and improvements

The questions are documented in the following:

- **Transparency, concepts and functionalities**
  - Transparency is the capability of enabling local and remote resources to be accessed using identical operations. Rate and describe how PLEDGER achieves transparency and a smooth transition to edge/cloud resources.
  - Which functionalities do you consider as the most useful ones?
  - Is there any functionality not considered within the system that you see as a 'nice-to-have' one?
- **Operation and execution of experiments**
  - Could you correctly execute your experiments?
  - Are your experiments dynamic in the use of resources or they always use the same ones?
  - Did you detect any limitation in terms of productivity (i.e. problems caused by Pledger in the pilot)?
- **Support and collaboration**
  - Was the documentation easy to understand and reliable?
  - Did you receive fast and helpful support?
- **Usability of the system**
  - How easy was the integration of your UC with Pledger components?
  - Was the deployment scheme provided satisfying in terms of performance?
  - In general terms, did you find PLEDGER easy to use?
- **Results and improvement**
  - Rate your improvements with respect to previous technologies used (1-5)
  - Do you think you would have been able to achieve the same results without using PLEDGER?
  - Do you have any suggestion for improvement?