



## D5.8 Pledger overall validation and evaluation

Document Identification			
<b>Status</b>	Final	<b>Due Date</b>	30/11/2022
<b>Version</b>	1.0	<b>Submission Date</b>	30/11/2022

<b>Related WP</b>	WP5	<b>Document Reference</b>	D5.8
<b>Related Deliverable(s)</b>	D5.3, D5.4, D5.7	<b>Dissemination Level (*)</b>	PU
<b>Lead Participant</b>	FILL	<b>Lead Author</b>	Verena Stanzl
<b>Contributors</b>	ATOS, ICCS, ENG, i2CAT, HOLO, INTRA, INNOV	<b>Reviewers</b>	Nikos Kapsoulis (INNOV) Orfeas Voutyras (ICCS)

<b>Keywords:</b>
evaluation, requirements validation, pilot evaluation, TRL

This document is issued within the frame and for the purpose of the Pledger project. This project has received funding from the European Union's Horizon2020 Framework Programme under Grant Agreement No. 871536. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

The dissemination of this document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains. This deliverable is subject to final acceptance by the European Commission.

This document and its content are the property of the Pledger Consortium. The content of all or parts of this document can be used and distributed provided that the Pledger project and the document are properly referenced.

Each Pledger Partner may use this document in conformity with the Pledger Consortium Grant Agreement provisions.

(\*) Dissemination level: **PU**: Public, fully open, e.g. web

## Document Information

List of Contributors			
Name	Partner	Name	Partner
Verena Stanzl	FILL	Clayton Gordy	HOLO
Francesco Iadanza	ENG	Roi Sucasas	ATOS
Gabriele Giammatteo	ENG	Nikos Kapsoulis	INNOV
Estela Carmona	i2CAT	Alexandros Psychas	ICCS
August Betzler	i2CAT	Orfeas Voutyras	ICCS
Bruno Cordero	i2CAT		

Document History			
Version	Date	Change editors	Changes
0.0	19/10/2022	Verena Stanzl (FILL)	ToC released. Responsible partners assigned.
0.1	20/10/2022	Verena Stanzl (FILL)	Changed naming and grouping of sections in Section 4.1.
0.2	02/11/2022	Verena Stanzl (FILL), Francesco Iadanza (ENG)	Added inputs for Sections 4.1.1 and 4.1.2. Added tables for KPIs in Section 4.3 and objectives in Section 6.
0.3	15/11/2022	Verena Stanzl (FILL), Estela Carmona (i2CAT), August Betzler (i2CAT)	Added inputs for SOE and RAN Controller in Section 4.1.1.
0.4	15/11/2022	Verena Stanzl (FILL), Roi Sucasas (ATOS), Nikos Kapsoulis (INNOV), Gabriele Giammatteo (ENG)	Added inputs for Orchestrator, SLA Lite, Blockchain, and Benchmarking evaluation in Section 4.1.1, 4.1.3, 4.1.4, 4.1.7 respectively.
0.5	17/11/2022	Verena Stanzl (FILL), August Betzler (i2CAT), Estela Carmona (i2CAT), Clayton Gordy (HOLO)	Added inputs from the UCs in Section 5.
0.6	21/11/2022	Verena Stanzl (FILL)	Added inputs for Section 2 and 3. Introduction added.
0.7	25/11/2022	Verena Stanzl (FILL)	Updated inputs in Section 2, 4.1. Acronyms added. First version of Executive Summary and Conclusions provided.
0.8	28/11/2022	Verena Stanzl (FILL), Alexandros Psychas (ICCS)	Updated Section 4.1.2.2. Added Section 6. Updated Section 4.3. Updated Executive Summary and Conclusions.
0.9	29/11/2022	Verena Stanzl (FILL), Lara Lopez (ATOS), Nikos Kapsoulis (INNOV)	Updated Section 2 based on feedback. First internal review. Corrections included. References added.
0.10	29/11/2022	Verena Stanzl (FILL), Gabriele Giammatteo (ENG), Clayton Gordy (HOLO), Orfeas Voutyras (ICCS)	Updated Section 4.3 and 5.1-Completed Section 4. Final internal review.
0.11	30/11/2022	Carmen San Román (ATOS)	Quality assurance review

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation	<b>Page:</b>	2 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b>
			Final

Document History			
Version	Date	Change editors	Changes
1.0	30/11/2022	Lara López (ATOS)	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Verena Stanzl (FILL)	30/11/2022
Quality manager	Carmen San Román (ATOS)	30/11/2022
Project Coordinator	Lara López (ATOS)	30/11/2022

# Table of Contents

---

Document Information .....	2
Table of Contents .....	4
List of Tables.....	6
List of Figures .....	7
List of Acronyms.....	8
Executive Summary .....	10
1 Introduction .....	11
1.1 Purpose of the document.....	11
1.2 Relation to other project work.....	11
1.3 Structure of the document.....	11
2 Methodology.....	12
3 Validation of requirements .....	13
4 Pledger system evaluation .....	15
4.1 Metrics of individual assets and integration.....	15
4.1.1 Orchestration Subsystem– Orchestrator, Recommender, SOE, RAN Controller.....	15
4.1.2 Configuration Subsystem – ConfService, AppProfiler.....	20
4.1.3 Benchmarking Subsystem – Benchmarking Suite.....	22
4.1.4 SLAs Subsystem – SLA Lite.....	23
4.1.5 Blockchain Subsystem – SLASC Bridge.....	24
4.1.6 Security Subsystem – T&R Engine and other mechanisms.....	25
4.1.7 Cross-subsystems evaluation – Automated service orchestration .....	26
4.2 Report on TRL, IRL, and SRL.....	27
4.3 Technical KPIs.....	31
5 Use Case evaluations.....	33
5.1 Use Case 1: Mixed Reality applications on the edge .....	33
5.1.1 Dynamic load allocation and resource management .....	33
5.1.2 Multi-User Remote Service Support and Training .....	33
5.1.3 Pledger Evaluation and User Feedback .....	34
5.1.4 Secure encryption of confidential data .....	34
5.2 Use Case 2: Edge infrastructure for enhancing safety of VRUs .....	34
5.2.1 Evaluation of the risk detection and notification system for VRU safety .....	34
5.2.2 Infrastructure Management.....	35
5.2.3 Management of a critical service.....	35
5.2.4 Management of sensitive information .....	36
5.2.5 Pilot evaluation and end users' feedback .....	37
5.3 Use Case 3: Manufacturing the data mining on the edge.....	41

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation	<b>Page:</b>	4 of 69	
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	
	<b>Version:</b>	1.0	<b>Status:</b>	Final

5.3.1	Thermal and process stability of manufacturing process.....	41
5.3.2	Automated resource management to ensure a smooth shopfloor process .....	41
5.3.3	Management of sensitive information .....	43
5.3.4	Pilot evaluation and end users' feedback.....	44
6	Mapping outcomes to objectives .....	47
	Conclusions .....	51
	References .....	52
	Annex .....	53

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	5 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

## List of Tables

---

Table 1: Evaluation metrics of Orchestrator.	15
Table 2: Evaluation metrics of DSS.	16
Table 3: Evaluation metrics of SOE.	17
Table 4: Evaluation metrics of RAN Controller.	19
Table 5: Evaluation metrics of ConfService.	20
Table 6: Hardware configurations for UC2 and 3 for App Profiler assessment.	20
Table 7: App Profiler evaluation results.	21
Table 8: Evaluation metrics of Benchmarking Suite.	22
Table 9: Evaluation metrics of SLA Lite.	24
Table 10: Evaluation metrics for SLASC Bridge.	24
Table 11: Evaluation metrics for T&R Engine.	25
Table 12: Average Satisfaction (%) table for different types of networks.	26
Table 13: Evaluation metrics for automatic service orchestration.	27
Table 14: Technology Readiness Level definitions.	27
Table 15: Integration Readiness Level definitions.	28
Table 16: The Pledger IRL matrix.	29
Table 17: Pledger components and their TRL and Component SRL.	29
Table 18: System Readiness Level descriptions and matching to Composite SRL.	30
Table 19: Achievement of the projects' KPIs based on produced results.	31
Table 20: UC2 refund policies.	36
Table 21: UC3 refund policy.	42
Table 22: Mapping of outcomes to Pledger high level objectives.	47
Table 23: List of initial functional requirements and their validation.	54
Table 24: List of initial non-functional requirements and their validation.	63
Table 25: List of additional functional requirements and their validation.	68
Table 26: List of additional non-functional requirements and their validation.	69

## List of Figures

---

<i>Figure 1: Alert notification gadget for UC2 mounted on an electric scooter. The green LED indicates readiness of the application and on-street radio connectivity.</i>	37
<i>Figure 2: Experimenters and UC2 team at the tram station observing how the scooter approaches.</i>	38
<i>Figure 3: Experimenters filling out the questionnaires after participating in the pilot.</i>	38
<i>Figure 4: The machine during the warm-up phase</i>	42
<i>Figure 5: Snapshot of Pledger wallet with accessed information about parts produced in a given timeframe</i>	43
<i>Figure 6: The Cybernetics UI in the shopfloor.</i>	45

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	7 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

## List of Acronyms

Abbreviation / acronym	Description
3D	Three Dimensional
5G	5 <sup>th</sup> generation of cellular technology
API	Application Programming Interface
AR	Augmented Reality
CAD	Computer Aided Design
CI/CD	Continuous Development/ Continuous Integration
CPU	Central Processing Unit
DApp	Decentralised Application
DLT	Distributed Ledger Technology
DoA	Description of Action
DSS	Decision Support System
Dx.y	Deliverable number y belonging to WP x
E2CO	Edge-To-Cloud Orchestrator
EC	European Commission
FaaS	Function as a Service
FR	Functional Requirement
GB	GigaByte
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HTT(S)	Hypertext Transfer Protocol (Secure)
HL2	HoloLens 2
IDS	Intrusion Detection System
IRL	Integration Readiness Level
KPI	Key Performance Indicator
LED	Light Emitting Diode
ML	Machine Learning
MR	Mixed Reality
Mx	Project Month x
NFR	Non-Functional Requirement
OBU	On-Board Unit
PET	Privacy Enhancing Technology
PLC	Programmatic Logical Controller
PoC	Proof of Concept
REST	REpresentational State Transfer
Q&A	Questions & Answers
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network

Abbreviation / acronym	Description
RDNS	Risk Detection Notification System
SLA	Service Level Agreement
SOE	Slice Orchestration Engine
SSL	Secure Sockets Layer
SRL	System Readiness Level
SSH	Secure Shell protocol
T&R	Trust & Reputation
TLS	Transport Layer Security
TRL	Technology Readiness Level
UC	Use Case
UI	User Interface
V2X	Vehicle-To-Everything
VRU	Vulnerable Road User
WP	Work Package

## Executive Summary

---

The present deliverable handles and documents all the activities undertaken under the context of Task T5.4 “*Validation and overall evaluation*” to validate the Pledger components and solutions and evaluate their benefits in different domains provided by the project’s Use Cases.

The Pledger system is evaluated in several aspects:

- ▶ Validation of the requirements to ensure the correctness and completeness of the system.
- ▶ Evaluation of the individual components in terms of correctness and consistency.
- ▶ Determination of the corresponding Technology, Integration, and System Readiness Levels.
- ▶ Evaluation of Pledger functionalities through the Pledger UCs.
- ▶ Project evaluation in terms of objectives and KPIs.

The pilot evaluations form the centerpiece of this deliverable. Pledger functionalities are evaluated in three domains: a) Augmented Reality in industrial environments, b) Micromobility in Smart City environments to increase the safety of Vulnerable Road Users, and c) process stability on the manufacturing shop floor.

The UCs are demonstrated through pilots in the relevant environments and use Pledger functionalities, (such as service orchestration, performance monitoring, automated decision making, etc.) to improve the performance, flexibility, scalability, and security of the system. Furthermore, the reservation of infrastructure and network resources is demonstrated.

The involvement of end-users during the pilots made it possible to gather user feedback related to the Pledger solutions and UCs, which was throughout positive. Users appreciated the quality of the experience, as they notice no mentionable interruptions during the pilot. The use case applications were received as valuable and beneficial. However, some suggestions for future developments were given for further improvement of the use cases.

The individual components are assessed in terms of consistency and correctness and their Technology Readiness Level is determined. Furthermore, the Integration and System Readiness Level is assessed.

This activity showed that Pledger has achieved adequate maturity for all of its individual components and as a whole system.

To complete the overall picture of Pledger, the outcomes are mapped to objectives. Every functional component of Pledger actively contributes to the fulfilment of at least one high-level objective described in the Description of Action (DoA). Furthermore, the UCs development and evaluation contribute to the fulfilment of crucial objectives related to Quality of Experience (QoE) and Quality of Service (QoS).

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	10 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

# 1 Introduction

---

## 1.1 Purpose of the document

---

The aim of this deliverable is to provide a report of all the evaluation activities performed in the course of Task *T5.4 “Evaluation and Validation”*. The UC pilots are evaluated in a consistent and uniform way including the feedback of all stakeholders and users involved. Qualitative and quantitative feedback is gathered using surveys and metrics defined in the different UCs. Furthermore, the evaluation is extended to cover the Pledger system and its components themselves, so as to evaluate the correctness, consistency, and completeness of the Pledger solution as a whole. Finally, the project evaluation ends by mapping the outcomes to the initially defined objectives of the project.

## 1.2 Relation to other project work

---

This deliverable builds upon all technical WPs, starting with WP2, where all specifications to build and implement Pledger were defined. The evaluation is initiated with the requirements defined there, which are validated during the T5.4. Furthermore, the Key Performance Indicators (KPIs) defined in this WP are used to support the evaluation of the system, both in terms of reporting and in terms of methodology, making it possible to draft surveys to derive the associated feedback.

Furthermore, the evaluation includes the components implemented in WP3 and WP4 and their evaluation for correctness and consistency. Task *T5.2 “Integration and Demonstrator”* handled all integration work based on Pledger components to enable the establishment of Pledger and the use of its functionalities in the pilots.

Finally, the previous work performed in WP5 is used as a basis. In Task *T5.1 “Application development”* all UC-specific applications and all applications to make use of Pledger functionalities were implemented. These applications are evaluated during the pilot executions and reported as part of the overall evaluation. Task *T5.3 “Pilot operation and monitoring”* handled the integration of Pledger features in the UCs as well as the set-up of scenarios and experimentation and the definition of valuable metrics to use in this task. The outcomes of these pilot executions are also reported in this document.

## 1.3 Structure of the document

---

Beyond this point, this document is structured in six major chapters:

- ▶ **Chapter 2** presents the methodology applied in this deliverable ensuring Pledger’s overall evaluation.
- ▶ **Chapter 3** presents the validation of requirements to ensure the correctness and completeness of the system.
- ▶ **Chapter 4** reports the metrics and evaluations of the different Pledger components structured in sub-systems. Furthermore, the Technology Readiness Level (TRLs), Integration Readiness Levels (IRLs), and the System Readiness Level (SRL), as well as the technical KPIs are reported.
- ▶ **Chapter 5** evaluates Pledger through its UCs, where the different scenarios are quantified and the feedback of involved end-users is reported.
- ▶ **Chapter 6** maps all project outcomes to Pledger objectives.
- ▶ **Chapter 7** concludes this deliverable.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	11 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

## 2 Methodology

---

Task T5.4 “*Validation and Evaluation*” handles the evaluation of the UC pilots as well as the overall evaluation of the Pledger system. The evaluation and validation activities ensure the system’s correctness, completeness, and consistency. In this section, the overall approach and methodology applied is described.

The evaluation of the Pledger system took place in five major parts:

- ▶ **Validation of requirements:** At the beginning of the project, all requirements were collected from the different partners to build the basis for the architecture and implementation of the different tools. In M24 (i.e., after two years), they were updated, and the updates were reported in D5.3. During T5.4, this updated list was validated. Each requirement was assessed individually, by checking whether this requirement was fulfilled and where and how this was demonstrated or enabled. The positive validation of the requirements ensures that Pledger is complete and correct as a system and working as planned.
- ▶ **Evaluation of components:** The next step is the evaluation on subsystem level and the validation of the correct and consistent function of the components. For this purpose, the component owners defined specific procedures and performed them. This was done using either testing scenarios or, where applicable, using UCs and their applications. Furthermore, metrics from these procedures were extracted to support the evaluation and its documentation. Afterwards, some metrics for evaluation of the system are determined. These are especially focused on cross-subsystems procedures. The results of this activity are reported in Section 4.1.
- ▶ **Evaluation of TRLs, IRLs, SRL:** Readiness level of the Pledger components, of the integration points between them, and of the system as a whole are quantified in terms of TRL, IRL, and SRL respectively, following the methodology introduced in [11].
- ▶ **Evaluation of UCs:** The UC evaluation is the centerpiece of this deliverable to validate the Pledger approach through three UCs. The focus is set on evaluation of benefits of using Pledger functionalities. The procedures and scenarios tested are described in D5.7 [1]. In this deliverable, the focus is set on the involvement of end users and gathering their feedback regarding QoS and QoE of the components involved and functionalities offered by Pledger. This feedback is gathered via surveys during the pilot execution and described in detail in the UC subsections 5.1-5.3.
- ▶ To complete the project evaluation and to provide an overall picture, the technical KPIs set for the project are reported. Lastly, the outcomes of the project are mapped to the objectives stated in the Description of Action (DoA). Each objective is considered individually and a mapping of the major outcomes from all components is provided in a table. This concludes the overall validation and evaluation of Pledger in this deliverable.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	12 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

### 3 Validation of requirements

Following the methodology described above, the requirements are validated to ensure Pledger is complete and correct. At the beginning of the project, the requirements were collected, analysed, prioritised, and documented in the D2.2 [2].

After identifying the relevant stakeholders with the help of user stories, the requirements were formalised and prioritised using the MoSCoW technique [3]. An updated version of the requirements was given in the deliverable D5.3 [5].

In this deliverable, the actual implementation of these requirements is evaluated. After the successful validation and evaluation of the requirements' implementation, the Pledger system can be considered as correct – concerning correct functionalities for all involved users. Furthermore, it can be considered as complete, as all desired features and functionalities from the different stakeholders involved are fulfilled.

For this activity, all requirements were analysed individually by their owners, who checked whether and how each requirement was fulfilled.

Initially, 60 functional and 51 non-functional requirements were identified. During the second year, these requirements were reviewed and 7 additional functional and 4 additional non-functional requirements were reported in D5.3 [5], resulting in 122 requirements in total.

From these 122 requirements, 3 were dropped and 7 were partially done or changed and 112 requirements were successfully implemented and demonstrated. The status of the requirements and their validation, i.e., where this requirement is demonstrated or described are given in in Table 23 Table - Table 26 of the Annex. The most important points concerning dropped requirements are given below.

The use of machine learning in the UC3 was dropped (NFR.26), as during the development of the UC applications, the relations in the data were identified as not as complex as initially expected, obviating the need for machine learning. However, if the training is automatised and packaged as a container, the use of specialised hardware can be enforced in the deployment options of the service on ConfService. Furthermore, the requirement of having a dashboard (FR.16) showing which data has been released to whom, was dropped. The release of data is handled via the StreamHandler, and a configuration is set about which data is pushed to the StreamHandler. This enables the direct control of releasing data, which is an advantage.

FR.08 describes a requirement where applications using Function-as-a-Service could be deployed. During further development, this requirement was not considered as necessary anymore and therefore it was dropped. This requirement was initially ranked with “could”, therefore its implementation was not crucial to the partners and did not impair any development during the project.

All notifications in the project were handled over StreamHandler and REST API for real-time notifications, therefore enabling further options of notification channels was not necessary, and requirement FR.27 was changed accordingly.

The deployment options and suggestions based on machine learning algorithms (FR.29 and FR.79) were changed and modified. These requirements have been modified as not enough data was available for machine learning. The effort has been moved to design and implement Lagrangian optimisations and the successful implementation was demonstrated with DSS demos about ECODA, TTODA and EA-ECODA algorithms in D4.6 [4].

Two requirements about end-user devices were changed. First, the registration on services that implement the Vulnerable Road User (VRU) safety features (FR.38) was changed to use an interface to add specific users enabling only them to use the application. Hence, Pledger does not require any intervention with end-user devices as devices only interact with the applications, but not with the platform. Therefore, NFR.34 has been changed and no minimum interventions over end-user devices are required.

During the first two years of development, a further requirement to enable the orchestration of infrastructures using a distributed architecture arose (NFR.ADD.2). This requirement was not implemented

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	13 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

due to missing time and resources. However, this requirement was not crucial for the implementation of the project but represents an interesting requirement to extend the functionalities of the Orchestrator in the future and can be considered in a forthcoming project.

All requirements and their validation are provided in Table 23 - Table 26Table in the Annex. The validation shows, that 91.8% of the requirements has been implemented successfully, whereas 5.7% was partially implemented or slightly changed. The rest of the requirements were dropped for the aforementioned reasons.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	14 of 69		
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0	<b>Status:</b>	Final

## 4 Pledger system evaluation

After validating the requirements, to ensure the correctness and completeness of the system, the Pledger system is evaluated. First, the individual components are evaluated to ensure their correctness, and consistency of the components and metrics, where applicable, is extracted and measured. Where possible and meaningful, applications and scenarios from the UCs were used to generate the results.

After the analytical assessment, the components and system are evaluated on a high level in terms of TRL, IRL, and SRL. Furthermore, the technical Key Performance Indicators (KPIs) for this project are assessed, and their status is reported.

### 4.1 Metrics of individual assets and integration

To evaluate the individual components, applicable metrics are defined and assessed. The different component owners were responsible for this activity, and were tasked with the evaluation of the different components and inclusion of the UCs in the procedure, where applicable and useful. In the following subsections, these results are presented following the structure of subsystems identified in the Pledger architecture (reported in D2.3 [6]).

#### 4.1.1 Orchestration Subsystem– Orchestrator, Recommender, SOE, RAN Controller

The orchestration subsystem is responsible for the reservation of computational and network resources and the management of containerised applications. It acts as the link between the Decision Support System (DSS) and the underlying infrastructure. The Orchestrator, DSS, Slice Orchestration Engine (SOE) and Radio Access Network (RAN) Controller belong to this subsystem and are described in the following.

##### 4.1.1.1 Orchestrator

The Edge-To-Cloud Orchestrator (E2CO) component is responsible for handling the deployment and management of the Pledger applications based on the instructions and actions received from the DSS.

This component was running in the Pledger testbed (allocated at ENG premises) with an uptime value of 99% during the last period of the project where all the final tests had been made. During this time, the E2CO component has deployed and managed applications in six different infrastructures: Docker [7], Kubernetes [8], MicroK8s [9], K3s [10], SOE, and a custom VM (UC1). The metrics are summarised in Table 1.

Table 1: Evaluation metrics of Orchestrator.

Orchestrator		
Metric	Purpose	Result
Availability	Measure the percentage of time E2CO is up and running in the Pledger infrastructure / testbed.	99% E2CO container running on Kubernetes cluster in ENG testbed.
# of types of infrastructure engines	Measure the number of target infrastructures supported by the Orchestrator.	6 Docker, Kubernetes, MicroK8s, K3s, SOE, and custom VMs

##### 4.1.1.2 DSS

The goal of the DSS component is to find the best node to deploy applications, while providing different optimisations and supported scenarios to Service Providers. This is implemented as a Monitoring, Analysis, Planning, and Execution (MAPE) loop.

A major part for the functionality of the DSS is the time spent to process key metrics and trigger a DSS decision. As the DSS is designed to trigger actions asynchronously, with control threads running every 60 seconds monitoring key metrics, this process lasts 60s max. This includes the following key metrics: requested and reserved resources (Central Processing Unit - CPU, memory), POD startup times, node-to-node latency and SLA violations received. Monitoring the DSS container running on Kubernetes cluster in ENG premise results in 99% availability. This is due to the fact, that new releases are deployed through Kubernetes with rollout update and internal load balancer, so without any disruption. The accuracy of DSS actions is taken as 100%, except for the EA-ECODA algorithm, where accuracy for resource threshold is 90% due to approximation done on the validation spreadsheet. Table 2 summarises the metrics.

Table 2: Evaluation metrics of DSS.

DSS		
Metric	Purpose	Result
Time spent to process key metrics and trigger a DSS decision	Time spent by the DSS to detect key metrics value changes and SLA violations and to take a decision about scaling/offloading.	60s max
Availability	Measure the percentage of time the DSS is up and running in the Pledger infrastructure.	99%
Correctness	Accuracy of the DSS actions taken.	100% for all DSS algorithms, except for EA-ECODA (90%)

#### 4.1.1.3 SOE

SOE is responsible for the configuration and deployment of end-to-end network slices including compute, network, and radio chunks, as well as a radio service. SOE manages the deployment of some network slice components, namely compute chunks, network chunks and the set of software components (radio service) that enable the radio connectivity. In addition, SOE integrates with the RAN Controller for the deployment and configuration of radio chunks with the required radio devices.

In Pledger, all the above-mentioned configurations are performed automatically upon request, from the ConfService. When a network slice is provisioned through Pledger, the orchestrator sends to SOE all the required API calls for the configuration of each of the individual slice components, their grouping into a single slice entity, and the activation of the components required by the radio service.

The ability to perform these configurations in a fully automated manner bring two distinct advantages. First, the configurations can be completed in a much shorter time than a manual deployment, and can be executed by a non-skilled technician, whereas manual configurations require a lengthy process that must be carried out by a highly skilled engineer. The second advantage is that the automated deployment reduces the scope for human error, as users only need to provide a few initial configuration parameters.

In order to evaluate the time required for both manual and automated deployments, manual deployments have been executed by an experienced user for both Vehicle-To-Everything (V2X) and 5G slices. The time required for each step has been measured and compared to that required by SOE when using Pledger. Results are summarised in Table 3. It was demonstrated that automated configurations through Pledger provide a 7-fold and 21.9-fold improvement for V2X and 5G slices, respectively, with respect to the time required for manual configurations. Table 3 gives a summary of all metrics mentioned.

Interested readers can find additional information about the pilots performed in Pledger, including both 5G and V2X slicing, in D5.7 [1].

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	16 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

Table 3: Evaluation metrics of SOE.

SOE		
Metric	Purpose	Result
# of actions supported	Both the provisioning and unprovisioning of network slices are supported. This provides additional flexibility to infrastructure managers.	2
# of infrastructures supported	Kubernetes and Openstack are supported as underlying technologies. This provides additional flexibility to infrastructure managers.	2
Types of network slices supported	Two types of network slices are supported: (1) slices based on Kubernetes infrastructures with IEEE 802.11p radios, and (ii) slices based on Openstack infrastructures with 5G NR. The SOE supports additional types of network slices, but these are not integrated nor showcased in Pledger.	23
Kubernetes-based compute chunk creation time (manually)	Time needed for the deployment of a compute chunk in a Kubernetes infrastructure, spanning the time required to perform all required steps manually.	60s
Kubernetes-based compute chunk creation time (Pledger)	Time needed for the deployment of a compute chunk in a Kubernetes infrastructure, spanning the time since request until the compute chunk is fully available.	<1s
Openstack-based compute chunk creation time (Manually)	Time needed for the deployment of a compute chunk in an Openstack infrastructure, spanning the time required to perform all required steps manually. Note that this is one of the processes that occur during the deployment of an end-to-end network slice.	180s
Openstack-based compute chunk creation time (Pledger)	Time needed for the deployment of a compute chunk in an Openstack infrastructure, spanning the time since request until the compute chunk is fully available. Note that this is one of the processes that occur during the deployment of an end-to-end network slice.	2s
Kubernetes-based network chunk creation time (manually)	Time needed for the manual deployment of the required networking elements over a Kubernetes infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end network slice.	180s
Kubernetes-based network chunk creation time (Pledger)	Time needed for the automated deployment (using Pledger) of the required networking elements over a Kubernetes infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end network slice.	1s
Openstack-based network chunk creation time (manually)	Time needed for the manual deployment of the required networking elements over an Openstack infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end network slice.	240s

SOE		
Metric	Purpose	Result
Openstack-based network chunk creation time (Pledger)	Time needed for the automated deployment (using Pledger) of the required networking elements over an Openstack infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end network slice.	4s
V2X radio service deployment time (manually)	Time needed for the manual deployment of the V2X stack software elements over a Kubernetes infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end V2X network slice.	210s
V2X radio service deployment time (Pledger)	Time needed for the automated deployment (using Pledger) of the V2X stack software elements over a Kubernetes infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end V2X network slice.	48s
5G radio service deployment time (manually)	Time needed for the manual deployment of the 5G stack software elements over an Openstack infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end 5G network slice.	240s
5G radio service deployment time (Pledger)	Time needed for the automated deployment (using Pledger) of the 5G stack software elements over an Openstack infrastructure. Note that this is one of the processes that occur during the deployment of an end-to-end 5G network slice.	9s
5G slice creation time	Time needed for the end-to-end deployment and activation of a network slice with 5G new-radio elements, since the provisioning request is activated in Pledger. This includes the time needed by Racoon to deploy a radio chunk, as well as the times needed by SOE to deploy a compute chunk, a network chunk, and a radio service.	57s
V2X slice creation time	Time needed for the end-to-end deployment and activation of a V2X slice with IEEE 802.11p radio elements, since the provisioning request is activated in Pledger. This includes the time needed by Racoon to deploy a radio chunk, as well as the times needed by SOE to deploy a compute chunk, a network chunk, and a radio service.	61s

#### 4.1.1.4 RAN Controller

The RAN Controller is responsible for configuration and management of radio devices of an infrastructure. On one side, the RAN Controller configures physical parameters, such as radio frequency or transmission power of radio interfaces, on the other side it configures the radio services, i.e., the logic that provides connectivity to users and connects the radio with the remaining resources of a slice (networking, computation).

These configurations are all performed automatically upon request, i.e., upon the press of a button. Compared to a manual configuration, this saves a lot of time. Moreover, with failsafe mechanisms the RAN Controller implements, potential configuration errors are avoided that could lead to a malfunctioning of the radio infrastructure. This is a key point, given the complexity of the solutions and the many intermediate steps that are taken. Furthermore, the fact that the process is automatized also eliminates the need for experts to perform configuration tasks. Note that the RAN Controller uses a variety of software tools and protocols to perform its tasks, such as OpenDaylight, OpenvSwitch, OpenFlow, and NETCONF. Two radio technologies have been integrated and demonstrated in Pledger, namely IEEE 802.11p and 5G NR. Other technologies, such as 4G or Wi-Fi are supported by the RAN Controller but have not been integrated during the project.

For the evaluation of the benefit of using the RAN Controller with all its features to configure a radio infrastructure compared to a manual configuration, we measure the time it takes to reserve a radio chunk (i.e. a set of physical radio resources that form part of a slice) and to instantiate the radio service that provides connectivity to the user equipment with the services deployed in the slice. The times for the manual deployment have been measured for an experienced user who has sufficient knowledge to perform all necessary configurations, i.e., someone with deep technical knowledge.

Two experiments are done: one for the configuration of a single IEEE 802.11p radio (necessary for the deployment of V2X slices), and one for a 5G cell. Note that in case more than a single radio is configured, the manual deployment takes N times longer (N = number of radios to be configured), whereas the automatized deployment via Pledger takes only a few seconds longer for each device.

As shown in Table 4, a considerable improvement is observed, e.g., reducing the setup time from 590 seconds in the case of cellular technologies to barely 34 seconds, or even more, from 480 to 10 seconds, in the case of V2X radios. Note that the radio configuration time can be split into radio chunk creation time and radio service creation, which is detailed further below in Table 4.

Additional information about the pilots performed in Pledger, including 5G and V2X slicing, can be found in D5.7 [1].

**Table 4: Evaluation metrics of RAN Controller.**

RAN Controller		
Metric	Purpose	Result
# of actions supported	Both the provisioning and unprovisioning of radio services are supported. This provides additional flexibility to infrastructure managers.	2
# of radio technologies supported	5G NR and IEEE 802.11p are supported. Additional radio technologies are supported too, but those are not integrated nor showcased in Pledger.	2
Time to deploy V2X radio (manually)	Measure manual configuration time of a single V2X radio.	480s
Time to deploy V2X radio (Pledger)	Measure automatic configuration time of a single V2X radio.	10s
Time to deploy 5G radio (manually)	Measure manual configuration time of a single 5G cell.	590s
Time to deploy 5G radio (Pledger)	Measure automatic configuration time of a single 5G cell.	34s

#### 4.1.2 Configuration Subsystem – ConfService, AppProfiler

The Configuration subsystem is responsible for storing the infrastructure and application configuration that is managed either manually by the IaaS/ SaaS providers or by tools that support automatic discovery features. ConfService and AppProfiler belong to this subsystem.

##### 4.1.2.1 ConfService

ConfService supports the Service Provider with 9 different DSS optimisation algorithms options and their configurations available through the ConfService User Interface (UI).

Availability is 99% when monitoring the ConfService container running on Kubernetes cluster in ENG premise, as new releases are deployed through Kubernetes with rollout update and internal load balancer, so with no disruption. Both metrics are summarised in Table 5.

Table 5: Evaluation metrics of ConfService.

ConfService		
Metric	Purpose	Result
Number of DSS optimisation options	Measure the number of different DSS optimisation algorithms that can be configured in the ConfService.	9
Availability	Measure the percentage of time the ConfService is up and running in the Pledger infrastructure.	99%

##### 4.1.2.2 App Profiler

The assessment of the App Profiler was performed using Use Cases 2 and 3 applications. In particular, the dataset required to measure the component's KPIs was established by profiling the two use case applications in heterogeneous hardware configurations. For both of these instances of applications, two different hardware configurations were used in order to showcase the hardware agnostic behaviour of the component. Specially, for Use Case 2, different levels of workload were set and fed to the app so that an evaluation of the predictions could be done. In more detail, for Use Case 2 and 3 the two hardware configurations that were used are presented in Table 6.

Table 6: Hardware configurations for UC2 and 3 for App Profiler assessment.

Use Case 2 Hardware Configurations (200 Profiles extracted)			
CPU	Disk	RAM	OS
2 vCPUs	50 GB	4GB	Ubuntu 20.4
Intel Quad-core processor J4115	50GB	32GB	Ubuntu 20.4
Use Case 3 Hardware Configurations(200 Profiles extracted)			
CPU	Disk	RAM	OS
Intel Core i5-6442EQ	480GB SSD	16GB	Debian 4.19
Intel Core i5-6442EQ	480GB SSD	4GB	Debian 4.19

For UC2, 5 different stress levels were used, for both configurations, which correspond to 1, 5, 10, 15, and 20 users using the application. For each different combination of hardware configuration and stress level, the application was profiled 20 times, resulting in a total of 200 profiles. Respectively, for UC3, 100 profiles were produced, resulting again in a total of 200 profiles.

It is important to mention that the hardware configurations were as diverse as possible, with different amounts of resources and hardware component architecture. After successfully obtaining the testing data

from the UC applications, the evaluation process was conducted. App Profilers' evaluation is compromised by two evaluation types: Firstly, there is the time sensitivity of the processes of App Profiler, more specifically the classification model training time and profile classification time evaluation. Secondly, accuracy and overall consistency of the classifications produced by the component were evaluated. All the metrics used, their purpose, as well as the results are presented in Table 7.

Table 7: App Profiler evaluation results.

App Profiler		
Metric	Purpose	Result
Time of Training	Measure the time that is needed to train a new classification model.	9742 ms / Model(training Dataset 20000 profiles)
Time of Classification	Measure the time that is needed to classify an application profile.	1306ms/1000 application profiles 0.1306ms/ 1 application profile
Accuracy/Correctness of predictions	Measure the accuracy of the predictions made for an application.	92% (400 profiles from UC2 and UC3 applications)
Consistency of results	Measure the percentage of classification consistency for a specific application.	93.33% (400 profiles from UC2 and UC3 applications)

As far as the analysis of the metrics evaluation results is concerned, the findings for every metric are analysed below:

- ▶ **Time of Training:** Time of training is a crucial metric for the feasibility and sustainability of the component, given the fact that the component will need retraining to incorporate in the model new benchmark profiles. Usually, the training of new models is a time-consuming procedure, but through the usage of ML algorithms the time needed was significantly lower than a typical AI based solution (i.e., Neural Networks). For testing purposes, a significantly bigger dataset was created containing 20000 profiles. These profiles were pushed to the ML training algorithm, the training time even with a much larger data set was less than 10 seconds, strongly indicating the substantial speed of the training process.
- ▶ **Time of classification:** Even though typically ML/AI classification system do not require much time to classify data, this test was performed in the context of App profilers' evaluation completeness. As it was expected App Profiler was very fast. To be exact, with a batch of 1000 profiles it needed 1.3 seconds to complete all the classifications.
- ▶ **Accuracy/Correctness of predictions:** For every ML based prediction/classification algorithms, one of the most predominant evaluation criteria is the accuracy. For measuring the accuracy of the whole process, the 400 profiles that arose from the two Use Cases were split in Training and Testing datasets with a 70/30 percentage. The split was not made completely random, but with respect to each configuration and workload level combination. The resulting training dataset was then appended to the Default Model of the profiler and a new model was trained. The remaining profiles, which comprise the testing dataset, were given as input to the profiler and the resulting predictions were then evaluated as far as the accuracy is concerned. The actual accuracy of predictions for these profiles was 92%. It is important to mention, that this evaluation is tightly related to a Technical KPI of the project (Technical KPI 11: Accuracy of classification of Use Case applications to benchmark categories). Given that the success criteria of this Technical KPI was >90% accuracy the 92% measured accuracy surpasses the success threshold.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	21 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

- ▶ **Consistency of results:** For each combination of hardware configuration and workload, the consistency percentage is defined as the predominant prediction among the multiple predictions. Consistency of value is an important part of the correctness evaluation, because although the algorithm can be accurate, it can produce many varying classifications depending on the nature of workload introduced to the application at the time of profiling. The resulting consistency of 93% showcased a very good stability for the component.

#### 4.1.3 Benchmarking Subsystem – Benchmarking Suite

The Benchmarking subsystem provides performance data of configured infrastructures to better characterise them and to optimise the orchestration with suggestions on application performance. The Benchmarking Suite is the only tool in this subsystem.

For the Benchmarking Suite, two types of metrics have been monitored to evaluate the component:

- ▶ performance metrics (e.g., availability, reports calculation time) measured on monitoring data collected in the last three months of project;
- ▶ quantitative metrics about the usage of the component in Pledger (e.g., number of tests executed, number of infrastructures assessed) measured on data generated by the Benchmarking Suite in the last year of project.

Metrics have been measured on the Benchmarking Suite instance deployed in the ENG testbed and used by all the project use cases. All metrics are documented in Table 8.

From the measurements collected, it is possible to see how the number of tests and performance metrics collected only over the last year of project is remarkable (more than 27.2k executions and 2.8 million measures). This allowed also validating the robustness and scalability of the Benchmarking Suite tool.

The operation that takes more time is the generation of benchmarking reports (27s on average). It corresponds to the total time to generate updated performance assessment reports for all benchmark tests, for all infrastructures' nodes and for all the users of the Pledger instance used in the project, and notify them and the other components through the StreamHandler. The number of the reports generated is 344 (measured on 16<sup>th</sup> November, 2022) with very minor variations during the last year of project (mainly caused by tests and demos done in the infrastructure). A summary of all metrics is given in Table 8.

Table 8: Evaluation metrics of Benchmarking Suite.

Benchmarking Suite		
Metric	Purpose	Result
# of workloads defined	The number of workloads that the Benchmarking Suite can simulate during the test executions.	57 from 14 different tools
# of assessed infrastructures	The number of different infrastructures against which tests have been executed. Some infrastructures include multiple nodes and/or flavours tested.	6
# of executions *	The total number of test executions for all benchmarks and all infrastructures.	27,298

Benchmarking Suite		
Metric	Purpose	Result
# of metrics collected*	The total number of metrics collected. Each test execution generates multiple metrics.	5,574,792 In addition, 2,348,776 system metrics have been collected (CPU, memory, network) during test executions
Availability of Benchmarking Suite service**	The availability of the Benchmarking Suite services.	Scheduler: 99% Model Synchroniser: 98.1% Reports Generator: 98.1%
Model Sync time**	Time necessary to sync the Benchmarking Suite configuration with the entire Pledger model (all users, infrastructures and projects).	Average: 14s 75 <sup>th</sup> percentile: 14.3s 95 <sup>th</sup> percentile: 14.5s
Analytics calculation time**	Time necessary to calculate aggregated metrics from all the raw data. They are calculated once a day and stored.	Average: 1.53s 75 <sup>th</sup> percentile: 1.48s 95 <sup>th</sup> percentile: 4.79s
Reports generation time**	Time necessary to calculate performance assessment reports for all infrastructures, tests and users configured in Pledger. Reports generation is executed daily.	Average: 27s 75 <sup>th</sup> percentile: 30.5s 95 <sup>th</sup> percentile: 30.9s

(\*) calculated for about the last year of project: from October 21<sup>st</sup>, 2021 to November 9<sup>th</sup>, 2022

(\*\*) calculated over the last three months of project: from September 1<sup>st</sup>, 2022 to November 25<sup>th</sup>, 2022

#### 4.1.4 SLAs Subsystem – SLA Lite

The SLA Lite component is responsible for doing the assessment of the SLAs managed by Pledger. It relies on a monitoring engine application which is continuously gathering metrics from the different infrastructures (i.e., Kubernetes testbed), and it does the correspondent assessment. If the SLA Lite detects a violation, then it sends a notification to the DSS component via Kafka.

The result of monitoring this containerised component running on Kubernetes cluster in ENG premise is 99%. During this time, the accuracy of the violations detected and notified is 100%. As this component relies on Prometheus instances (via the Monitoring Engine / StreamHandler), once the metrics are stored there, the SLA Lite just evaluates the values to do the SLA assessment. This means that this accuracy depends on other components, i.e., Prometheus.

Finally, the time the SLA takes to detect a violation and notify it, is approximately 60s. The SLA Lite gets the metrics from the monitoring components every minute, does the evaluation of these metrics, and in the case of a violation, a notification to the other Pledger components is sent via Kafka. All metrics are summarised in Table 9.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation	<b>Page:</b>	23 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b>
			Final

Table 9: Evaluation metrics of SLA Lite.

SLA Lite		
Metric	Purpose	Result
Availability	Measure the percentage of time the SLA Lite is up and running in the Pledger infrastructure.	99% SLA Lite container running on Kubernetes cluster in ENG testbed.
Correctness	Accuracy of the violations detected and notified.	100%
Reliability	Time spent to detect and notify a violation in SLA guarantees.	~ 60s

#### 4.1.5 Blockchain Subsystem – SLASC Bridge

The Blockchain subsystem contains tools offering features such as smart contracts and refund schemes. Particularly, SLASC Bridge materialises the representation of SLAs contractual terms on the blockchain environment and handles any emitted SLA violations with the adopted on-chain compensation scheme. The component completes the SLA to smart contract process that produces a corresponding agreement's logic into contractual terms equivalents on the ledger and offers the decentralised paradigm of SLAs between providers and adopters.

Moreover, SLASC Bridge accomplishes the SLA violation handling process by automating the refunding of the SLA parties. Particularly, in case of an SLA breach, the deployed compensation scheme is triggered on the ledger and the management of the user accounts balances occurs accordingly.

SLASC Bridge operates the created smart contracts over a trustworthy network that functions securely in open and trustless edge infrastructures, while employing seamless integration and high automation in code triggering and execution in such environments. Table 10 reports all evaluation metrics for SLASC Bridge.

Table 10: Evaluation metrics for SLASC Bridge.

SLASC Bridge		
Metric	Purpose	Result
Compute time from SLA configuration to smart contract creation	Average time from SLA configuration to Smart Contract creation for different applications running on the Pledger infrastructure.	14s The total average time from SLA configuration to Smart Contract creation depends on the time needed for the processes of the creation of the SLA in the ConfService, the streaming time through StreamHandler, and the smart contract creation time of SLASC Bridge.
Availability	The percentage of time the SLASC Bridge is up and running in the Pledger infrastructure.	99%

SLASC Bridge		
Metric	Purpose	Result
Consistency	The 1:1 representation of the SLA configurations on the DLT.	1:1 Each SLA configuration that is created and deployed on the ConfService is matched to exactly one (1) smart contract structure (equivalent) that automatically deploys on the blockchain network.
Correctness	Level of data integrity and trustworthiness.	100% for all the deployed SLA configurations' smart contract equivalents on the DLT. Blockchain immutable nature guarantees data integrity and trust among the submitted transactions on the ledger.

#### 4.1.6 Security Subsystem – T&R Engine and other mechanisms

The Pledger Trust & Reputation (T&R) Engine transforms data related to the SLAs monitoring and SLAs violations into indexes like Trust, Reputation, and Popularity, to facilitate the selection of potential providers based on an extra criterion: their trustworthiness, the confidence that can be placed on them for the successful delivery of a specific service. For the model to be deemed successful, it has to be tested in different types of networks in which several nodes will have different types of malicious behaviours. The related metrics are presented in the following table.

Table 11: Evaluation metrics for T&R Engine.

T&R Engine	
Metric	Purpose
Responsiveness to dynamic behaviour	Nodes may dynamically enter, exit, and re-enter the network. This kind of behaviour may be followed in some specific types of attacks, like the whitewashing one, through which, malicious nodes attempt to get rid of their low reputation.
Responsiveness to oscillating behaviour	Nodes may change their behaviour from malicious to benevolent and vice versa. In the first case, the “social re-integration” of the nodes has to be achieved, while in the second case, the “social exclusion” of the nodes has to be performed in a prompt manner.
Responsiveness to collusion	Depending on the scenario, malicious nodes may work with each other to increase their T&R indexes. In the case of Pledger, the only such index that can be falsely presented is that of the Popularity of the node. The anomalies detection feature of the tool can identify and respond to such cases.
Responsiveness to complex behaviours	This refers to any combination of the above behaviours.

The Pledger T&R model was tested using TRMSim-WSN [12], the state-of-the-art simulator for T&R systems. The simulations were run for one service type. The Pledger T&R model was tested in multiple simulation environments where the nodes can have collusive, oscillating, and/or dynamic behaviour. The results are provided in the following table.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation	<b>Page:</b>	25 of 69	
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	
	<b>Version:</b>	1.0	<b>Status:</b>	Final

Table 12: Average Satisfaction (%) table for different types of networks.

% of malicious nodes/ behaviour type	10	20	30	40	50	60	70	80	90
normal	100	100	100	100	99	99	99	98	96
dynamic	100	99	99	98	97	95	93	91	77
oscillating	99	98	97	94	92	86	82	65	49
oscillating & dynamic	98	96	95	92	85	82	69	59	43
collusion	100	99	98	97	94	91	86	69	47
collusion & dynamic	100	99	97	91	91	85	79	62	45
collusion & oscillating	99	97	95	91	84	76	65	52	27
col. & dyn. & osc.	98	94	91	87	79	71	63	45	22

Regarding the rest of the Security mechanisms provided by Pledger (e.g., IDPS), they are presented in great detail in D4.1 and D4.4. As such, the already reported measurements will not be repeated in the present document.

#### 4.1.7 Cross-subsystems evaluation – Automated service orchestration

Lastly, two scenarios were evaluated, where more than one component is involved. They are focused on the management of applications, i.e., starting and stopping an application and offloading an application after an SLA violation. The time needed to start an application by the platform is max. 35s. In this procedure, the service provider needs to configure the possible infrastructures for the application and press the start/stop button in ConfService. The procedure is fully automatised and ConfService, DSS, Orchestrator, application and StreamHandler are involved. The metric is computed as follows:

- ▶ 5s max for the DSS to trigger the E2CO using the StreamHandler plus
- ▶ 5s max for the message exchange on the StreamHandler plus
- ▶ 5s max for the E2CO to trigger the App start/stop using the StreamHandler plus
- ▶ time needed for the App to start (App and infrastructure dependent, on the testbed Nginx takes 20s).

In a manual procedure, the service provider needs to access the docker host of the specific infrastructure, pull the image from a Docker registry with a docker command and afterwards run the specific image from the console. Depending on the experience of the software provider, this process lasts several minutes and is prone to errors (e.g. typing errors, pulling the wrong image, etc.). Apart from the fact that this procedure takes longer than the automated one, Pledger benefits cannot be exploited. For example, the DSS recommends the automatically the best suitable infrastructure for the application based on Benchmarking and App Profiler results.

Furthermore, the time needed to offload an application is determined. This time is measured as end-to-end time by the DSS to identify key metrics condition, where offloading is required and the time to execute the action resulting in 90 sec max. In this procedure, DSS, Orchestrator, Application and StreamHandler are involved and the metric is computed as follows:

- ▶ 60s max for the DSS to trigger the action (see DSS metrics in the “evaluation of components”) plus
- ▶ 5s max for the E2CO to trigger the App start/stop using the StreamHandler plus
- ▶ 5s max for the message exchange on the StreamHandler plus
- ▶ time needed for the App to start (App and infrastructure dependent, on the testbed Nginx takes 20s)

The manual procedure involves the stopping of the container on the current infrastructure, accessing the Docker host on the new infrastructure and start the container there and lasts several minutes. Again, Pledger features cannot be exploited in the manual procedure, as it is automatising the start/stop of the application as well as choosing the optimal infrastructure for the offload.

Table 13 summarises these two metrics.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation				<b>Page:</b>	26 of 69	
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0	<b>Status:</b>	Final

Table 13: Evaluation metrics for automatic service orchestration.

Cross-subsystems			
Metric	Purpose	Components involved	Result
Time needed to Start/Stop an App	Measure the end-to-end time needed by the platform to start/stop an App when the command is executed on the ConfService UI.	ConfService, DSS, E2CO, App, Stream-Handler	35sec max, with dependency on the actual App launched
Time needed to offload an App	Measure the end-to-end time needed by the DSS to identify key metrics condition where an action is required (offloading/scaling) and the time to execute the action.	DSS, E2CO, App, StreamHandler	90s max, with dependency on the actual App launched

## 4.2 Report on TRL, IRL, and SRL

In this section, the readiness level of the Pledger components, of the integration points between them, and of the system as a whole are quantified in terms of TRL, IRL, and SRL respectively, following the methodology introduced in [11].

The first step is the identification of the Technology Readiness Level (TRL) of the several Pledger tools and components, as they have been identified in D2.3 and D5.6. The purpose of the TRL is to measure the maturity of technology components for a system. The measurement allows an understanding of how much development a certain technology needs before being utilised. The TRL scoring is based on a scale from 1 to 9, with 9 being the most mature technology. Table 14 provides the definitions for the different levels of the scale.

Table 14: Technology Readiness Level definitions.

TRL	Definition/Description
9	Actual system proven through successful mission operations.
8	Actual system completed and qualified through test and demonstration.
7	System prototype demonstration in relevant environment.
6	System/subsystem model or prototype demonstration in relevant environment.
5	Component and/or breadboard validation in relevant environment.
4	Component and/or breadboard validation in laboratory environment.
3	Analytical and experimental critical function and/or characteristic Proof-of-Concept.
2	Technology concept and/or application formulated.
1	Basic principles observed and reported.

The TRL matrix of a system with n components is defined as a vector, where  $TRL_i$  is the TRL of component i:

$$[TRL]_{nx1} = \begin{bmatrix} TRL_1 \\ TRL_2 \\ \vdots \\ TRL_n \end{bmatrix}$$

The corresponding vector for the Pledger system is provided in the second column of Table 17.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation	<b>Page:</b>	27 of 69	
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	
	<b>Version:</b>	1.0	<b>Status:</b>	Final

The next step is the identification of the Integration Readiness Levels (IRLs) between components. The IRL approach is a systematic measurement of the interfacing of compatible interactions for various technologies and the consistent comparison of the maturity between integration points. In other words, for each integration point (pair of interacting components in the system) an IRL is identified. Table 15 provides the definitions for the different levels of the scale.

Table 15: Integration Readiness Level definitions.

IRL	Definition/Description
9	Integration is mission proven through successful mission operations.
8	Actual integration completed and mission qualified through test and demonstration in the system environment.
7	The integration of technologies has been verified and validated with sufficient detail to be actionable.
6	The integration technologies can accept, translate, and structure information for its intended applications.
5	There is sufficient control between technologies necessary to establish, manage, and terminate the integration.
4	There is sufficient detail in the quality and assurance of the integration between technologies.
3	There is compatibility (i.e., common language) between technologies to orderly and efficiently integrate and interact.
2	There is some level of specificity to characterise the interaction (i.e., ability to influence) between technologies through their interface.
1	An interface (i.e., physical connection) between technologies has been identified with sufficient detail to allow characterisation of the relationship.

The IRL matrix of a system with n components is defined as an nxn matrix where the integration between components i and j is represented by  $IRL_{ij}$ . The theoretical integration of a component i to itself is denoted by  $IRL_{ii}$  and is assumed to be a maximum:

$$[IRL]_{n \times n} = \begin{bmatrix} IRL_{11} & \cdots & IRL_{1n} \\ \vdots & \ddots & \vdots \\ IRL_{n1} & \cdots & IRL_{nn} \end{bmatrix}$$

The corresponding matrix for the Pledger system is provided in Table 16. This table is based on the integration points table already introduced in D5.6.

Table 16: The Pledger IRL matrix.

Pledger	Orchestrator	DSS	SasS/IaaS Monitoring Engine	Configuration DB	App Profiler	Configuration Dashboard	Benchmarking Scheduler	Bechmarks Metrics DB	Benchmarking Creator	SLA Manager	SLA Notifier	SLA Monitoring	SLASC Bridge	Anomalies Reasoner	Anomalies Rules	T&R Engine	SOE	RAN Controler
	Orchestrator	9	9	8	9	0	0	0	0	0	0	0	0	0	0	0	0	8
DSS	9	9	8	9	0	0	0	9	0	0	8	0	0	0	0	0	0	0
SasS/IaaS Monitoring Engine	8	8	9	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0
Configuration DB	9	9	0	9	9	9	0	0	9	0	0	0	0	0	0	0	0	0
App Profiler	0	0	0	9	9	0	0	0	8	0	0	0	0	0	0	0	0	0
Configuration Dashboard	0	0	0	9	0	9	0	0	0	0	0	0	0	0	0	0	0	0
Benchmarking Scheduler	0	0	0	9	0	0	9	0	0	0	0	0	0	0	0	0	0	0
Bechmarks Metrics DB	0	9	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0
Benchmarking Creator	0	0	0	0	8	0	0	0	9	0	0	0	0	0	0	0	0	0
SLA Manager	0	0	0	9	0	0	0	0	0	9	0	0	0	0	0	0	0	0
SLA Notifier	0	8	0	0	0	0	0	0	0	0	9	0	8	0	0	8	0	0
SLA Monitoring	0	0	8	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
SLASC Bridge	0	0	0	0	0	0	0	0	0	8	0	9	0	0	0	0	0	0
Anomalies Reasoner	0	0	0	0	0	0	0	0	0	0	0	0	9	9	7	0	0	0
Anomalies Rules	0	0	0	0	0	0	0	0	0	0	0	0	9	9	0	0	0	0
T&R Engine	0	0	0	0	0	0	0	0	0	8	0	0	7	0	9	0	0	0
SOE	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	8
RAN Controler	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	9

From the multiplication of the TRL vector and the IRL matrix, it is possible to extract a vector providing the **SRL for each individual component** (referred to as  $SRL_i$ ):

$$[SRL_i]_{1 \times n} = [IRL_i]_{n \times n} \times [TRL_i]_{1 \times n}$$

The  $SRL_i$  of a component in its turn, in combination with the number of the links of the component with other components, gives as a result the **Component SRL**. This is a metric of the maturity of a component both in terms of inherent TRL as well as its link to other components. Table 17 provides the corresponding values for the Pledger components and integration points.

Table 17: Pledger components and their TRL and Component SRL.

Component	TRL	# Links	Non-normalized $SRL_i$	Normalized $SRL_i$	Component SRL
Orchestrator	7	5	310	3.83	0.77
DSS	7	6	398	4.91	0.82
SasS/IaaS Monitoring Engine	7	4	247	3.05	0.76
Configuration DB	9	7	417	5.15	0.74
App Profiler	7	3	191	2.36	0.79
Configuration Dashboard	7	2	135	1.67	0.83
Benchmarking Scheduler	7	2	135	1.67	0.83
Bechmarks Metrics DB	9	2	137	1.69	0.85

Component	TRL	# Links	Non-normalized SRLi	Normalized SRLi	Component SRL
Benchmarking Creator	7	2	119	1.47	0.73
SLA Manager	7	2	135	1.67	0.83
SLA Notifier	9	4	233	2.88	0.72
SLA Monitoring	9	2	137	1.69	0.85
SLASC Bridge	7	2	135	1.67	0.83
Anomalies Reasoner	4	3	134	1.65	0.55
Anomalies Rules	7	2	99	1.22	0.61
T&R Engine	5	3	145	1.79	0.60
SOE	7	3	175	2.16	0.72
RAN Controller	7	2	119	1.47	0.73

The average of all Component SRLs provides the **Composite SRL of a system**. This metric gives an estimation of the maturity of the overall system being developed as a whole. The **SRL** is identified after the calculation of the **Composite SRL**.

Table 18 provides the definitions for the different levels of the SRL scale, and its mapping to the corresponding Composite SRL values.

Table 18: System Readiness Level descriptions and matching to Composite SRL.

SRL	Definition/Description	Composite SRL
5	Operation & Support	0.9 to 1.00
4	Production & Development	0.8 to 0.89
3	System Development & Demonstration	0.6 to 0.79
2	Technology Development	0.4 to 0.59
1	Concept Refinement	0.1 to 0.39

After doing the calculations, the Composite SRL for the overall Pledger system is 0.79. This value is mapped to an **SRL of level 3 – “System Development & Demonstration”**, while being at the very border of SRL4 – “Production & Development”. This indicates that the project, even though it is an R&D one, has achieved high levels of technical and integration maturity through the corresponding demonstrations and pilots, and thus is close to the production level of a solution.

### 4.3 Technical KPIs

To complete the project evaluation and to provide an overall picture of its technical results, the technical KPIs set for the project are reported and presented in the table below, together with their already achieved status for their fulfilment.

Table 19: Achievement of the projects' KPIs based on produced results.

KPI # and description	Target at the end of the project	Actual achievement
KPI1 – Pledger architecture blueprint <i>(Result 1)</i>	1 overall	Direct outcome of D2.3.
KPI2 – Pledger tools and methods for edge/cloud migration and decision support <i>(Result 2)</i>	5 tools	8 different tools are provided by Pledger: (1) App Profiler (ICCS), (2) Benchmarking Suite (ENG), (3) Config. Service (ENG), (4) DSS (ENG) + T&R engine (ICCS), (5) SLAs manager (ATOS), (6) Orchestrator (ATOS), (7) SLAs monitoring engine (ATOS).
KPI3 – Pledger SDN control framework to orchestrate dynamic network connections across distributed edge resources	1 overall	SOE and RAN Controller already using the SDN control (integration perspective provided in D5.6). A 5G demo is also completed.
KPI4 – Pledger privacy on edge <i>(Result 3)</i>	1 overall	Authentication, Authorization, and Encryption of the core StreamHandler tool.
KPI5 – Pledger Validated Use Cases <i>(Result 4)</i>	3 validated use cases	All three UCs have been validated in relevant and/or operational environments.
KPI6 – Improved experienced performance and stability of virtualised resources through enhanced provider resource management	>30% (measured through a series of benchmarks in an iterative manner and various testing scenarios)	48% on the side of provider selection optimisation (calculated on a set of different benchmarks representing different application types and observing performance improvements on infrastructures and nodes suggested by Pledger).
KPI7 – Number of supported SLA audits	>5	>6 (UCs and custom applications SLAs defined in ConfService; Available in D3.6, SLA database, and demos).
KPI8 – Number of supported Smart Contracts	>10	10+ (6 UC ones reported in D5.7 and 4 operational ones reported in D4.5. Furthermore, through the manifestation of the contractual terms coupling to smart contracts, the number is increasing accordingly.)
KPI9 – Number of blockchain frameworks deployed	>2 (at least 1 public and 1 permissioned)	2 different blockchain frameworks are deployed, and 3 different blockchain networks. For further elaboration, refer to deliverable D4.5.
KPI10 – Scalability of provider toolkit	>500 cores	Scalability of all components achieved, as present in D3.1-D3.6 and D4.1-D4.6.
KPI11 – Accuracy of classification of Use Case applications to benchmark categories	>90%	App Profilers' accuracy is at 92% (section 4.1.2.2 describes the experimentation and validation process).

KPI # and description	Target at the end of the project	Actual achievement
KPI12 – Improvement of the application running costs compared to current edge/cloud deployments	>20%	Directly linked to KPI6, resulting in savings from avoiding reimbursement costs.
KPI13 – Improvement of customer satisfaction from multiple available SLAs	Over 80% positive opinion in surveys	Achieved, as reported in Section 5.
KPI14 – Enhancement of understandability of edge/cloud metrics	Over 80% positive opinion in surveys	Achieved, as reported in Section 5.
KPI15 – Standardisation contributions with relation to SLA metrics and orchestration languages	Contribution to 2 standards	Direct contribution to 2 standards open-source initiatives for SLAs/DLTs, and to 3 standards landscape reports.
KPI16 – # of applications from the UCs	6+ (~2-3/UC)	Achieved, as presented in D5.5.

## 5 Use Case evaluations

---

In the following subsections, the evaluations of UCs are described. The evaluation consists of three parts:

- ▶ Evaluation of the UC applications
- ▶ Evaluation of the Pledger functionalities
- ▶ Evaluation of the pilot and users' feedback

Evaluating these three aspects gives an overall picture of the pilots and shows how UCs were improved both using the applications developed as well as with support of Pledger functionalities. Pledger functionalities also supported, during the development, the internal feedback from developers concerning functionalities for service providers and integration was gathered and described in deliverable D5.7. In the same deliverable, all procedures and scenarios to be evaluated were described.

This deliverable and especially this section describes the different problems solved in the specific UCs. The focus was set on the involvement of end-users and the improvements which can be brought to them. These aspects are presented in the following subsections.

### 5.1 Use Case 1: Mixed Reality applications on the edge

---

UC 1 sought to enhance the performance of augmented reality (AR) industrial service solutions through integration with Pledger edge computing infrastructures. A foundation of all mixed reality (MR) environments is in the visualization of, and subsequent interfacing with, three-dimensional (3D) computer-aided-designs (CAD). Such virtual interfacing should be of a considerable caliber, with high hologram resolution and correspondingly appropriate performance efficiency levels during real real-life hologram manipulations. To bypass current limitations in AR devices, namely their low level intrinsic computational processing levels and security vulnerabilities, the UC 1 application (PledgAR Workspace) combined with the Pledger toolkit was used to offload remote rendering abilities to an edge device and subsequently provide dynamic, real-time, decision making and resource allocations. Additionally, through the Pledger framework, encryption measures were used to add much needed security to the total encryption measures used during AR sessions. Collectively these Pledger-driven solutions aimed to enrich MR industrial tasks and simultaneously improve quality of experience levels for users. These solutions were assessed by specific experimental scenarios in this use case. The following sections will report on the results of these experiments, with emphasis on general contributions toward addressing the problems described above.

#### 5.1.1 Dynamic load allocation and resource management

Exploration into the extent of load allocations and resource management was assessed in real time during the pilot case scenario deployments. Across the fast prototyping test case, AR metrics were profiled. During the first tier of test case AR sessions, CAD files were loaded into the PledgAR Workspace and the time for loading of the files was recorded. Loaded CAD files were from Pledger partner FILL and represented machine items relevant to the local industrial environment. Multi-User Remote Service Support and Training

During the deployment of the UC1 multi-user service support and training, participants with no prior knowledge of an industrial device/process were connected with a remote expert who was able to sufficiently instruct them on the logistics and processes related to a device. This instruction was used to allow participants to execute some needed adjustment of the device. Remote instruction in these scenarios within AR space allowed successful training and instruction in this manner. Measurable results, in the form of feedback and responses from users, were collected.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	33 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

### 5.1.2 Pledger Evaluation and User Feedback

Qualitative support was assessed by questions posed to use case end-users (i.e., AR participant users). Overall, the performance of the different sessions were viewed favorably. Participants did (not) often notice substantial delays in latency or file loading times which would have impacted their session. During collaborative sessions in which multiple users were involved, latency metrics were similarly assessed. Collectively these data are interpreted as indicating a support in the efficacy of the PledgAR Workspace application. With respect to quality of experience, these metrics support a degree of satisfaction in maintaining quality of experience during AR sessions with PledgAR Workspace.

### 5.1.3 Secure encryption of confidential data

Encryption of data streams in these PledgAR-environments had no measurable metric but was nevertheless deployed in all sessions. As mentioned in previous deliverables within this project, the encryption architecture for the PledgAR Workspace application was utilized with Pledger’s distributed ledger technology (DLT) blockchain network. This framework allowed the generation of the custom-built Whisper protocol which contributed to secure double data encryption, specifically during the signaling phase described previously. The Whisper protocol was initiated during every signaling handshake event, i.e., during every launching instance of connecting the PledgAR Workspace edge device to the HoloLens 2 (HL2) client. In doing so, the Whisper protocol contributed to the overall encryption and security levels needed during the use case AR sessions and helped facilitate a sense of safety and security in MR environments. This was validated internally by the successful connection of the PledgAR Workspace edge to the client device

## 5.2 Use Case 2: Edge infrastructure for enhancing safety of VRUs

---

The execution of the UC2 risk detection and notification system (RDNS) pilot in Barcelona aiming to increase vulnerable road user (VRU) safety for micromobility scenarios has allowed for a series of evaluations to be performed that serve the validation of both application-specific aspects, but also Pledger-specific aspects. For the latter, a set of three particularly relevant features implemented by Pledger and used in the UC2 pilot have been identified: Pledger enables the ad-hoc configuration of a city-wide infrastructure and the creation of slices to serve specific applications at the “touch of a button”. Furthermore, Pledger assures that the QoS and QoE of the service are guaranteed by managing the critical service implemented by the RDNS application. Finally, the information gathered by the RDNS application from on-street devices (VRUs, tram) can be safely stored and only be accessed by trusted users via Pledger DLT. The DLT is also the component that allows for the creation of smart contracts to implement SLAs and refund policies.

The definition of the evaluation setup and configuration, as well as the steps taken during the experiments are defined in Section 5.4 of D5.7 [6]. The following subsections document the outcomes of the evaluations.

### 5.2.1 Evaluation of the risk detection and notification system for VRU safety

The experiment to evaluate the RDNS functionality was performed overall 13 times on-street. In these iterations, the correct operation of the on-board unit (OBU) hardware both for the tram and the scooters, the implementation of the RDNS application and whether the alert-gadget installed in the scooters does a good job in alerting the drivers was evaluated.

The pilot validation of the connectivity of the OBUs was successful, showing that the tram connects at 250m of distance to the station, while the scooter OBUs connect at 300m of distance to the station. This means that both types of OBUs are detected early by the application, allowing to assess potential risks with enough margin of time. Upon detection of a risky situation, it was checked whether the alerts to the scooter drivers were sent in time, i.e., when entering the warning/critical areas defined by the application. It was detected that in fact the alert is triggered immediately when the corresponding areas are entered. Minor deviation of the timings across repetitions was observed, resulting in the alarm to be sometimes triggered a few meters early or late. This deviation can be explained by the accuracy of the

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	34 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

GALILEO positioning used in the OBUs when having E1B/C signal. We measured the accuracy through a dedicated experiment and observed an average deviation of 1.6m. This deviation could potentially be reduced with more accurate, yet expensive Global Navigation Satellite System (GNSS) devices. Another factor that contributes to the accuracy drop is the GNSS coverage, which is known to suffer in cities, especially in the “street valleys” between rows of buildings.

The alerts were generated correctly by the OBU gadget, triggering off whenever one of the two areas was entered when a tram was at the station (no false-negatives), and not triggering off at all whenever there was no tram stationed (no false-positives). Feedback from users reveals that the gadget mounted on the scooters is loud and visible enough for the rider to perceive the warning to slow down and be cautious. In the moment the critical alarm triggers off, i.e., when the scooter is very close to the tram station, the acoustic signal can be also perceived by pedestrians in the tram stop area. Though this sound is not very loud, according to the feedback collected it can be perceived and makes pedestrians aware about the incoming scooter.

### 5.2.2 Infrastructure Management

Pledger fully automates the creation of end-to-end network slices made up of compute, network and radio elements, in a manner that is fully transparent to the infrastructure owner. In addition, Pledger allows for the configuration of some slice parameters, mainly related to the computational resources of the slices. For example, infrastructure owners can seamlessly configure the number of computational resources assigned to each slice. Other configuration parameters (e.g., those related to the radio elements) are pre-configured in Pledger and do not need to be adjusted by the user.

The time it takes to deploy a network slice through Pledger has been compared to that required for a manual deployment, where all the configuration steps are done manually by experts with advanced knowledge of the tools and process needed to perform the slice configuration. The slice creation time is a relevant metric, since, as the results show, a lot of time (up to 80%) can be saved with Pledger. In addition, by using Pledger, the slice deployment can be performed by less-skilled technicians, while still avoiding possible intermediate errors in the complex process of configuring the infrastructure. Another relevant benefit is that multiple isolated slices can be created on top of a single infrastructure, considering the QoS parameters required for each slice. The concurrent, yet isolated use of available resources allows for optimising resource usage across an infrastructure, which is a main goal for an infrastructure provider.

### 5.2.3 Management of a critical service

Pledger brings distinct advantages to service providers in general. In first instance, Pledger fully automates the deployment of services and applications in the target infrastructure, at the touch of a button, without the need for applying specific infrastructure-related configurations. This reduces time costs as well as the scope for human error. In addition, the service provider can easily apply relevant configurations, e.g., those related to the set-up of SLAs and DSS actions.

In addition, the management of UC2 critical service benefits from Pledger’s SLA and DSS functionalities as follows. Firstly, a number of previously configured SLAs can be activated through the ConfService; these include SLAs related to the instantaneous end-to-end delay and the average service availability. These SLAs serve a double purpose: (i) to trigger a DSS action such as an application scale-up, and (ii) to be used by the DLT module to calculate the smart contract balance according to some refund policy. These are explained in more detail below.

Critical services such as the RDNS in UC2 have certain QoS requirements that must be always met. In UC2, these are related to the end-to-end delay and the service availability. By ensuring that these requirements are met, the RDNS will be able to send out risk notifications in a timely manner, at all times, such that end users are notified of risky situations as needed. In Pledger, this is achieved by constantly monitoring the RDNS metrics through the Monitoring Engine. When the application incurs a large end-to-end delay (normally due to a large number of end users connected to the RDNS application), the SLA generates an alert, which is fed to the DSS; in turn, the DSS performs an application scale-up, increasing

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	35 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

the resource limits allocated to the application, which is then able to perform tasks in a shorter period of time, thus reducing the end-to-end delay to acceptable levels. The maximum end-to-end delay of road hazard signalling applications is defined as 1.5s by ETSI’s technical specification 101 539-1 for intelligent transport systems and V2X applications. In Pledger, several lower delay thresholds are defined, so that the DSS can take action whenever the delay starts increasing, before the critical value of 1.5 seconds is reached.

UC2 also benefits from the application of SLAs to Pledger DLT for the management of smart contracts. Should the application become unavailable or return a long end-to-end delay for a sustained period of time, an SLA violation message is generated and received by the DLT, which will apply a suitable refund policy depending on the criticality of the SLA violation. In UC2, there are two complementary refund policies. The first refund policy is based on the percentage of time over a 24-hour period in which the application end-to-end delay stays below 1.5 seconds. The second refund policy is based on the percentage of time that the application is available. In both cases, the SLA and refund policy are established as shown in the following table. Note that the values below have been selected to perform proof of concept and do not represent a specific business case.

Table 20: UC2 refund policies.

Accuracy refund		
Average time that delay stays below 1.5s over 24 hours (%)	Violation severity	Refund policy
Between 97% and 99%	Warning	$(1-0.985)*refund\_value$
Between 95% and 97%	Serious	$(1-0.965)*refund\_value$
Below 95%	Catastrophic	$(1-0.945)*refund\_value$
Availability refund		
Average application availability over 24 hours (%)	Violation severity	Refund policy
Between 97% and 99%	Warning	$(1-0.985)*refund\_value$
Between 95% and 97%	Serious	$(1-0.965)*refund\_value$
Below 95%	Catastrophic	$(1-0.945)*refund\_value$

#### 5.2.4 Management of sensitive information

Even though no personal data are collected by UC2 RDNS, some sensitive data are gathered, which are related to the users’ location at a specific time and to the likelihood of a risky situation. The collected information must be stored in a secure manner, such that it is only accessible to genuinely interested and trusted parties (e.g., Barcelona’s tram service organisation). In Pledger, this is achieved through the DLT module as follows.

Upon reception of incoming location-related messages from the OBUs to the RDNS, a log is generated which contains the positions of the tram and nearby OBUs, as well as information about any events that have been detected on-street, such as scooters facing a risky situation. Generated logs are periodically sent to Pledger’s Kafka server on the topic “vru\_positions”. The DLT receives these logs through a custom Kafka consumer, and then securely stores the data on the blockchain, ensuring that only users with sufficient permissions can retrieve the information.

### 5.2.5 Pilot evaluation and end users' feedback

In order to obtain feedback on the UC scenario and, in more generic terms, feedback on Pledger and its features, two iterations of the on-street use case scenario defined in D5.7 were performed involving experimenters who are not involved in the project. Before the pilot, a 20-minute presentation about Pledger and the UC was given, explaining the main features developed in the project and how they are applied in the UC. Using visual material gathered during the project and pre-pilot phase, the experimenters were shown what Pledger is capable to do, how it visualises information and how it helps to provide the best QoE possible to the users. Also, it was highlighted that the infrastructure to be used for the experiment was configured with the press of a button, reserving and configuring a dedicated slice across the city infrastructure just for the UC pilot.

After this introductory presentation and a Q&A and discussion session, the participants were invited to participate in the pilot experiment either by driving the scooter themselves, or by observing the execution of the experiment from the point of view of a pedestrian at the tram station. In order to simplify the experiment and to reduce possible risks of actual collisions, we created a virtual tram that announces its position to be at the tram station. This allowed for a much more flexible and fluent execution of the experiment that consisted in a scooter driver approaching the tram station and being alerted of a risk through an alert notification gadget (Figure 1).<sup>1</sup>



Figure 1: Alert notification gadget for UC2 mounted on an electric scooter. The green LED indicates readiness of the application and on-street radio connectivity.

In total, 7 experimenters drove the scooter, while 3 experimenters positioned themselves as pedestrians at the tram station (Figure 2). In each iteration, all experimenters were asked to pay special attention to the audiovisual signals emitted by the gadget when the warning and critical events were notified: whether the alarm beep was loud enough, the LED bright enough, the timing of the notifications prompt enough, and whether the differentiation between warning and critical was clear enough.

<sup>1</sup> We performed the pre-pilot evaluation and validation with an actual physical on-board unit to make sure it works.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	37 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final



Figure 2: Experimenters and UC2 team at the tram station observing how the scooter approaches.

Eventually, after all scooter riders had gone through the scenario execution, the experimenters were asked to fill out the questionnaire with questions about the UC and Pledger (Figure 3).



Figure 3: Experimenters filling out the questionnaires after participating in the pilot.

While all experimenters identified themselves as service consumers, two of them also fell into the service provider category. None of the experimenters was an infrastructure provider. We asked the experimenters three UC-specific questions:

- ▶ Whether they considered the experiment to be successful, i.e., whether all the promised features worked as expected;
- ▶ How they would rate the perceived QoE;
- ▶ If experimenters think that the same results could be achieved without Pledger and the features it implements.

All experimenters considered that the experiment was performed successfully and rated the QoE with an excellent 4.75 out of 5 points (1 = poor, 5 = excellent). Only one experimenter considered that the same results could be achieved without Pledger, and the rest agreed on the fact that Pledger implements key features which ensure the smooth QoE and successful execution of the Pilot.

The experimenters were also asked a series of additional, more generic questions. They were asked to rate:

- ▶ Their familiarity with Cloud/Edge and orchestration concepts;
- ▶ Their grasp of those concepts based on the Pledger presentation given before the experiment;
- ▶ Their grasp of the features and concepts introduced by Pledger;
- ▶ Whether metrics and information are exposed in an easy to read format by Pledger;
- ▶ How relevant they consider the Pledger features presented to them, and finally;

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	38 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

- ▶ How they evaluate the transparency of the cloud/edge migration and optimisation mechanisms applied by Pledger with the aim to maintain a high QoS and QoE.

It turned out that not all experimenters were familiar with the concepts of edge/cloud computation and orchestration; in a rating from 1 to 5 (1 = not familiar at all, 5 = expert), an average of 3.33 was observed, having some experts but also some experimenters with low (2) knowledge of these concepts. Fortunately, it was easy (4.3/ 5; 1=very difficult, 5 = very easy) for experimenters to understand the concepts after hearing the presentation given about them in the context of Pledger. The same level of understanding (4.3/5) was also perceived with regards to Pledger’s features and concepts.

It was also easy (4/5) to read the metrics and dashboards that expose information about the status of the infrastructure. The Pledger features were considered highly relevant (4.2/5) for the UC. Finally, a very high level of transparency (4.75/5) was perceived by the experimenters regarding the way that Pledger configures the infrastructure and manages the risk detection service, i.e., the mechanisms are not actively perceived by the users. Comments were also provided by experimenters with regards to transparency, with some examples<sup>2</sup> given below:

- ▶ *“Couldn’t notice any delay during the different iterations performed at Fluvità with an excellent transparency and a smooth transition to the edge/cloud resources, 5/5”*
- ▶ *“From a user perspective, completely transparent, 4/5.”*
- ▶ *“Totally transparent from a user perspective, 5/5.”*

The last two questions of the questionnaire were focused on the experimenters opinions and suggestions for possible improvements or features that could potentially be interesting for UC2 or Pledger in general:

- ▶ Which features used in the pilot do you consider most useful?
- ▶ Do you have any suggestions for improvements?
- ▶ Which Pledger features not available at the moment would you consider nice-to-have?

The experimenters listed the following features to be the most useful ones:

The scaling functionality in case of scenarios with lots of users.
Smart placing/scaling of applications.
The acoustic warning sound while approaching a critical area.
All the functionalities are useful to bring a more safety road to vulnerable road users.
The easy deployment of [the] system (slice, service, maintaining the QoS).
Low latency, resource distribution [(refers to slicing)] and orchestration/adaptation [of the service].

Apart from the useful features developed as part of the UC2 application to warn about the risky situation, experimenters highlight several of Pledger’s features as useful: The smart scaling and placement to ensure the QoE/QoS and the slicing.

<sup>2</sup> Some comments have been translated from Spanish/Catalan to English; similar comments provided in the answers to the questionnaire are only shown once here. Comments in brackets [] added by August Betzler.

Regarding possible improvements (both for the UC application and Pledger) the following comments have been provided:

Increase the volume of the critical alarm, it was too low.
Consider the velocity [of a VRU] for detecting risk situations and [the] degree of risk.
The beep when entering the critical zone should be improved, as it isn't heard as loudly as the warning zone one.
Expand the scope of the use case, e.g., by adding more locations or different risk situations.
The way to scaling the solution, based on physical equipment, maybe is a pain [in the sense of difficult].
To better define the area of warnings/critical for the application. [For this, user the warning/critical alert lasted too long.]

All the feedback provided focuses on the application and way the RDNS works. Some users complained about the beep not being loud enough. The loudness is adjustable, but it was chosen to reduce the volume to not alert unnecessarily pedestrians not involved in the pilot, and due to health concerns because the UC2 team from i2CAT was working with the beeping sound for many days in a row. Also, on the second day of the pilot, one of the buzzers stopped working properly, which contributed to this perception. Another comment pointed out that the solution applied is difficult to scale up in an infrastructure, which is true, considering public space is used. However, this is just one of the ways the RDNS can be implemented; there might be others that do not require dedicated infrastructure or even have more radio coverage. Another experimenter pointed out that it would be interesting to refine the application with a feature that considers not only the proximity to the tram station for deciding on the risk level, but also the speed of the scooter. This could be implemented, thanks to the vehicular stack used that also reports the velocity. Finally, a user complained that the areas for warning/risks should be better defined. For the pilot, a simplistic approach (i.e., two rectangles laid over the tram station, one for warning, one for critical) was followed. The vehicular stack allows to move these areas or to use freely shaped polygons.

Finally, feedback was also provided on which features would be nice-to-have but which are not shown in the pilot:

A user-friendly Dashboard with for example a dropdown [menu] to choose the application to deploy, or an intent engine that can translate a natural language query into low level operations.
When entering a danger zone, the scooter velocity should be automatically reduced ensuring the user safety.
Add some features that increase the possible usage of the RDNS, e.g. a virtual rear-mirror.
An additional display in the scooter that indicates the state / performance of the platform.

Even though we obtained only limited feedback on this point, some improvements were suggested. On the platform level, one user suggested a more user-friendly dashboard with some quality of usage improvements. This proposal comes from experience in another project where some additional features are present in the dashboard. The rest of suggestions focused on the risk detection application. One difficult to implement feature is the automated breaking when driving too fast when close to the tram station. This feature depends on regulations and laws and could be implemented as a proof of concept on a mechanical level, but falls into the category of automatic drive assistance, which currently does not exist for scooters or bicycles. Another experimenter proposed the implementation of additional features based on the available technologies, e.g., a virtual rear-mirror (e.g., a beeping or blinking) if another user approaches quickly from behind and intends to overtake. This indeed could be implemented using the vehicular stack used in UC2. Finally, a user thought it would be interesting (for the pilot execution) to have a screen on the scooter displaying some key performance metrics of the infrastructure.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation	<b>Page:</b>	40 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b>
			Final

### 5.3 Use Case 3: Manufacturing the data mining on the edge

During the execution of the UC3 pilot, a small manufacturing line was set-up and the experiments were performed, as defined in deliverable D5.7. This enabled the overall validation and evaluation of both UC applications and Pledger-specific aspects. Besides monitoring and determining the process and thermal stability of the manufacturing process, the QoS and QoE of the applications developed within the UC were assessed and improved using Pledger functionalities. These include the monitoring of performance measures and the automated resource management assuring the QoS and QoE guaranteed. Furthermore, sensitive information about parts produced during the pilot is stored safely on the DLT to ensure only authorised access to this information. Furthermore, the DLT is used for the creation of smart contracts implementing SLAs and refund policies, which are presented. All scenarios performed are described in the deliverable D5.7. The following subsections contain the outcome and main observations of the pilot evaluation.

#### 5.3.1 Thermal and process stability of manufacturing process

The experiments to evaluate the functionalities of the applications were performed in 3 consecutive steps. They gave the opportunity to evaluate the correctness of the applications as well as the benefits they provide to the operators on the shopfloor.

Firstly, the thermal-stability component was used to determine the thermal stable window of the machine specifying the threshold required for the thermal-stability indicator to be exceeded to automatise the whole warm-up process. Process experts expected the duration for the warm-up of one shift (i.e., 8 hours). Observations of the data showed that, after 5 hours, the thermal-stability coefficient reached a stable phase, and the threshold was set. This results in the saving of 3 hours which can already be used for actual production and subsequent increase in efficiency of 37.5%. Furthermore, two parts were measured, one part at the start of the warm-up (produced by the cold machine) the other at the established end of the warm-up. The first part did not pass the quality check and was classified as reject, whereas the other part satisfied the quality requirements. This shows that the stability of the thermal-stability indicator also correlates with the quality of the parts produced and is a reliable tool to reduce unnecessary unproductive times. Furthermore, the thermal-stability indicator is determined after downtimes during manufacturing and based on the result, the warm-up process is triggered, or the production is continued. During these experiments, also the influence of the temperature in the shop floor was observed. The influence was higher than originally expected and already short downtimes ~10 minutes (under the specific conditions) resulted in decreased thermal-stability coefficients and made warm-up necessary to avoid scrap parts.

After the warm-up, the manufacturing of parts is initiated. The results of the parts are observed right afterwards on the Cybernetics UI. The velocities varied in different sub-processes of the parts and the results of the analytical service parts-produced were obtained. The comparison of two parts with different cycle times is as expected; the part with reduced velocity during milling also shows longer duration of the milling sub-process. The involved participants found the UI and the options to compare different aspects useful and very intuitive.

#### 5.3.2 Automated resource management to ensure a smooth shopfloor process

The UC benefits from the full automatization of the deployment process, replacing several manual operations and speeding up the process of bringing applications into production. Furthermore, relevant configurations for SLAs and DSS actions can be implemented in ConfService to benefit from the functionalities.

During the first experiments, different response times to the Programmatic Logical Controller (PLC) of the machine are tested to establish a dedicated threshold until the response is too long for the user. To ensure a smooth warm-up process, 3 PLC cycles of waiting time resulting in 90ms were considered as feasible. A further cycle affects the experience of the end user in the shop floor in such a way, that they

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	41 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

notice the machine is waiting in idle mode and operators might wonder whether the machine is working as intended or not.

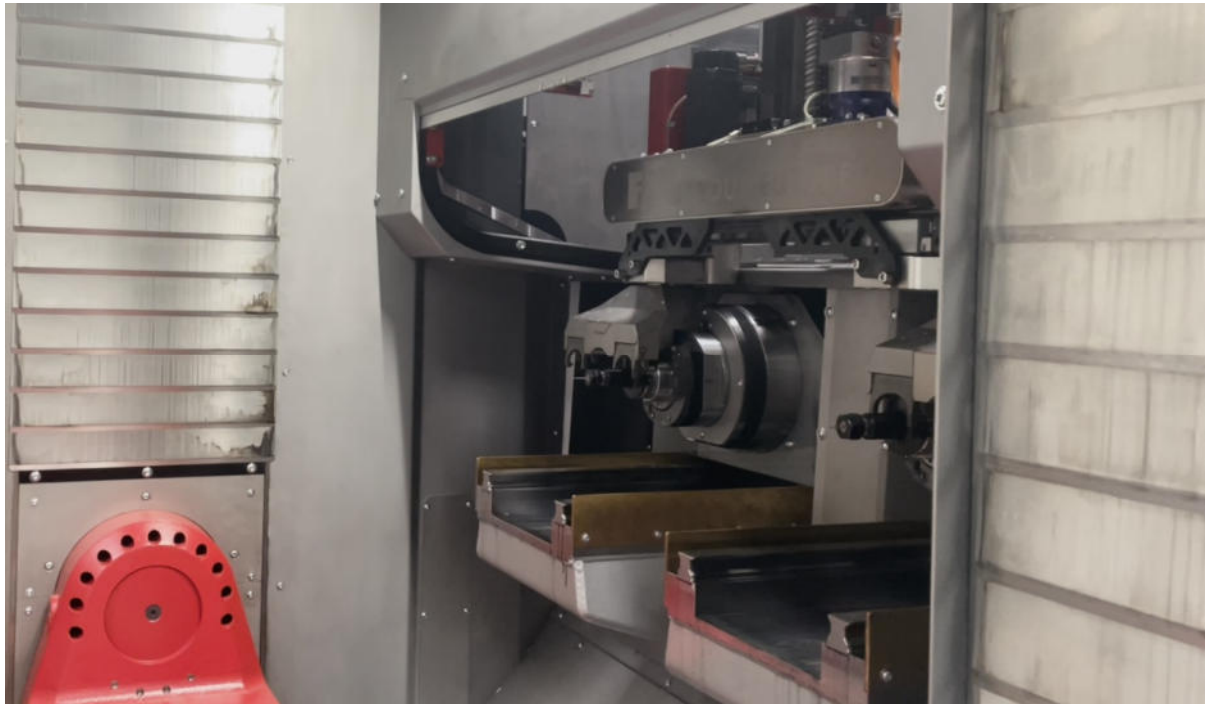


Figure 4: The machine during the warm-up phase

Using Pledger Monitoring Engine, the calculation time is constantly monitored. The same applies to the infrastructure metrics to monitor the behaviour of the infrastructure and machine. To guarantee the critical value of 90ms is never reached, the SLA threshold is set to 80ms and, in case the metric reaches the threshold, the DSS triggers the offload of another application to free resources to the thermal-stability component. The offload is useful because the thermal-stability component is not always under load, only when the warm-up procedure is done in the shopfloor, or the thermal-stability is determined after a short downtime. In case the warm-up is done for several hours and also the other analytical applications are running, the infrastructure is under load and the specified SLA thresholds might be violated. To reduce this risk, one of the analytical applications is offloaded and, after the successful warm-up, the analytical application is moved back to the edge.

As guaranteeing an optimal QoE for the end user on the shopfloor is critical, smart contracts can be used to establish a refunding policy in case the guarantees are not met for a certain amount of time. Depending on the severity of the SLA violation, a different refunding policy is applied. This does not represent a business case, only the technical feasibility is demonstrated. In the post-project phase, however, this can be evaluated and turned into a business case. The summary of the values and the refunding policy is described in the following. Note, that refunding in the project is defined as one unit and not counted against real money.

Table 21: UC3 refund policy.

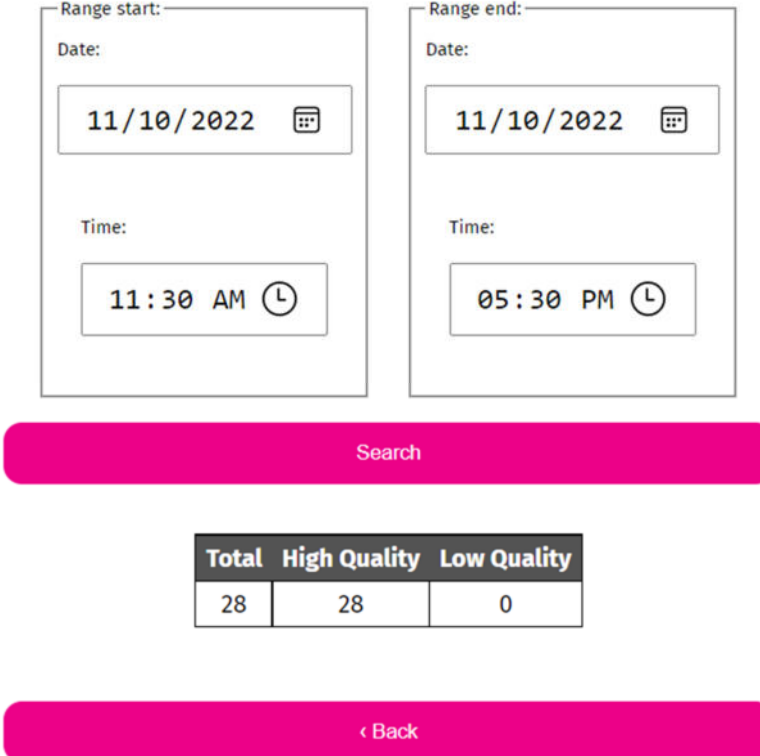
Average time that variance of calculation time is above X over 8hrs (%)	Violation severity	Refund policy
Between 95% and 99%	Serious	$(1-0.98) \cdot \text{refund\_value}$
Below 95%	Catastrophic	$(1-0.95) \cdot \text{refund\_value}$

### 5.3.3 Management of sensitive information

Some of the information related to the parts produced are considered sensitive, especially the amount produced in a given timeframe as well as the number of rejects. However, the information about the duration of sub-processes, e.g., loading, clamping, milling, unloading, the cycle time, and differences to target cycle time are crucial to analyse the process stability. During the project, the necessary data to be kept on the edge, and the necessary information to be transferred to the cloud were evaluated. All information related to the parts produced by the machine is stored in the database on the edge, where the Cybernetics UI can access it. As Cybernetics is also running solely on the edge device, only skilled personnel at the customer site with the necessary access information can access the information. However, the number of parts produced, and the number of rejects is also of special interest to specific persons at FILL, enabling them to establish future business models based on pay-per-use. Customers are not willing to share this specific information because they feel uncomfortable sharing and storing this information along with competitors. Therefore, the use of the Pledger DLT was evaluated to check whether this increases the trust of data transfer to the cloud for potential end users.

Four interviews with internal sales and projecting personnel with direct customer contact were performed, and results showed that all of them see potential for future innovative business models. Furthermore, three of them agreed, that the use of blockchain technologies with authorised access to the number of parts produced increases the trust of transferring data to the cloud. Especially, the fact, that the number of parts cannot be manipulated from both parties and that no third party can access the data was especially highlighted.

An example of accessing this information from the Pledger DLT is given in Figure 5.



The screenshot shows a search interface with two input boxes for 'Range start' and 'Range end'. Each box has a 'Date' field with a calendar icon and a 'Time' field with a clock icon. Below the inputs is a pink 'Search' button. The results are displayed in a table with three columns: 'Total', 'High Quality', and 'Low Quality'. Below the table is a pink '< Back' button.

Total	High Quality	Low Quality
28	28	0

Figure 5: Snapshot of Pledger wallet with accessed information about parts produced in a given timeframe

#### 5.3.4 Pilot evaluation and end users' feedback

During the pilot execution and evaluation, 7 end-users were involved (FILL internal personnel, but external to Pledger). Colleagues from the software department, mechanical engineering, and project management were involved. The feedback was gathered using Microsoft Forms in a digital way. The survey was sent out after the pilot and included some explanation and visualisation for their reference.

To interpret the results, they were asked to which stakeholder they relate. Most of them identified themselves as service consumers (5), one identified themselves as infrastructure provider, whereas 4 identified themselves as service providers. Multiple answers were possible.

Involved end-users were differently familiar with the concepts of Cloud/Edge computing, and Virtualisation and Orchestration of services. On a range of 1-5, two users voted 3, 4, and 5, respectively. One user was less familiar with these concepts and voted 2, resulting in an average voting of 3.71.

Before going into the pilot area, they were informed about Pledger and the highlights of the functionalities as well as the UC. The users understood the mentioned concepts after the presentation very well and it was easy for them to understand them, as the average result was 4.43. Given the average voting before and after the presentation, this resulted in an increase of 83% of better understanding of the concepts covered in Pledger. This is also reflected in the answers to the follow-up question, where it was asked how easily the main benefits of the system were understood. This question got an excellent voting of 4.86, showing that the Pledger benefits were easily understandable to the involved end-users. Moving on the metrics and measured QoS and infrastructure in the UC, the average voting is 4.2. The questions raised during the presentation and the feedback allow conclusions to be drawn that it was not very clear that the metric “calculation time” is highly tied to latency in the UC. Only one service consumers outside the software department was not aware of this fact, but could be clarified afterwards. This metric is more relevant to service providers, and service consumers should – in best case – not even notice any influences in this metric. Therefore, it was not considered as critical the fact that non-software experts could not relate much to this metric. However, it is pleasing that users considered the features of Pledger introduced in the UC highly relevant (4.43 out of 5).

Afterwards, questions tied to the UC and its applications were asked. For the evaluation of the project, it was especially important to gather feedback of process interruptions and the easiness to determine the process stability.

As the pilot was running for several days, participants were free to join the shopfloor individually based on their time schedule. Therefore, participants were active during the pilot at different times and the observation times were diverse, without the need for a user to participate several hours.

The participants were asked to monitor and observe the warm-up process, whether it goes smoothly or they experience any interruptions. The same applies for the manufacturing. Furthermore, they were asked to use the UI (see Figure 6) to determine the process stability and analyse any occurring irregularities.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	44 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final



Figure 6: The Cybernetics UI in the shopfloor.

This process was accompanied individually, using the individual dialogue as feedback channel. Additionally, three questions were asked to help quantifying the results. Firstly, feedback about interruptions experienced was asked. Furthermore, the understandability of the UI and the trust in the data transfer was asked with a rating from 1 to 5, where 5 was the highest option. A special focus was set on Stream-Handler and the fact, that only applications with a valid certificate can consume message. This concept was considered very transparent and the trust in data transfer was evaluated average with 4.14, which was higher than originally expected by the project team.

Furthermore, no specific interruption was experienced. Once, a participant reported an unexpected downtime due to some hydraulic issue, which was not caused by the software. The overall QoE was rated in average with 4.57 of 5, resulting in a satisfaction of 91.4%. The understandability of the UI to explore the parts produced and media consumption as well as the results of the thermal-stability indicator were rated as excellent, with an average of 4.71. Some improvements were suggested, mainly related to the selection of the colour scheme to also include visual-impaired people or small adaptations related to information in tool-tips (e.g., adding the reason for a reject part).

Lastly, the increase in trust using the Pledger DLT feature was evaluated separately. This was described in Section 5.3.3.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation	<b>Page:</b>	45 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU
		<b>Version:</b>	1.0
		<b>Status:</b>	Final

Summarising the outcomes from the UC3 pilot evaluation, the results are excellent. No specific interruption was experienced and users also reported that the short interruption between the times the machine performs the warm-up run and the decision (whether to continue the warm-up or go into production mode) is of reasonable duration and no specific concerns were raised. It is very gratifying that the concepts and benefits of the system were easy to understand despite the different prior knowledge of the participants. A little surprising, but very positive, were the results for trust in the data transfer and the increase in trust using the Pledger DLT. This opens up promising possibilities for future developments and innovations in the post-project phase.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	46 of 69		
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0	<b>Status:</b>	Final

## 6 Mapping outcomes to objectives

In the process of validation and evaluation of Pledger, it is important to include how the high-level objectives of the project guided the development and functionalities of the system, and by extension, every individual component. For this reason, a mapping of the high-level objectives and the main outcomes was created and delivered in Table 22 below. It is important to mention that these outcomes can be functionalities of Pledger components that directly address or aid the fulfilment of these objectives; UC implementation and development also played a crucial role in addressing crucial objectives (#4, #6).

Table 22: Mapping of outcomes to Pledger high level objectives.

Objective	Outcome
#1 - Enhance edge/fog computing provider resource management practices that lead to improved stability of offered services and QoS/QoE for end users.	<ul style="list-style-type: none"> <li>▶ <i>App Profiler</i> improves indirectly the resource management by providing to the DSS important information about the resource needs of the applications as well as classifications of agnostic (to the provider) applications, thus adding valuable information to the multi-criterial decisions about application deployment.</li> <li>▶ <i>SLA Lite</i> does a continuous assessment of the Pledger SLAs. When a violation is detected, it notifies the DSS to take the correspondent actions.</li> <li>▶ <i>SOE</i> and <i>RAN Controller</i> allow for the configuration of deployment of network slices with the required edge/fog computing resources that are necessary to meet given QoS/QoE constraints. In addition, resources allocated to a slice are always guaranteed to that slice, providing enhanced stability of QoS/QoE to services.</li> <li>▶ <i>Benchmarking Suite</i> provides performance assessments that can be exploited to improve the allocation of applications on an infrastructure optimising in such a way the usage of resources by keeping/improving the desired QoS.</li> <li>▶ The <i>DSS</i> implements smart orchestration algorithms to support scaling and edge-cloud-edge offloading to optimise resource consumption, latency and energy consumption on the edge, following the service provider's preferences based on the hardware availability on the edge.</li> </ul>
#2 - Architect and implement the coupling of edge/fog computing with blockchains and distributed ledger technologies.	<ul style="list-style-type: none"> <li>▶ <i>DLT</i> has directly built a trusted network that is able to operate securely in open trustless edge/fog infrastructures with facilitation of smart contracts implementation. Also, the component enables the deployment of decentralised applications (DApps) as a new paradigm of applications for the supported use cases of Pledger.</li> <li>▶ <i>SLASC Bridge</i> implements the automated coupling of SLA contractual commitments with smart contracts and their terms breaches handling when applicable, by employing high automation in executing code portions on the edge, while hiding the provider-specific details and technology-specific differentiations through the smart contract equivalents deployments and the automated refunding schemes.</li> </ul>
#3 - Enhance Cloud/Edge services behaviour measurability and predictability in order to increase operational trust, thus enabling their use in more critical	<ul style="list-style-type: none"> <li>▶ <i>App Profiler</i> aids the predictability and measurability of the Cloud Services through the monitoring and classification of the resource footprint of a variety of applications. The profiler's classifications</li> </ul>

Objective	Outcome
<p>applications and increase the synergies between cloud/edge infrastructures.</p>	<p>of known benchmarks, combined with the results of the benchmarking framework analysis, result in a tailored analysis of the infrastructure that takes into account the application needs.</p> <ul style="list-style-type: none"> <li>▶ <i>Benchmarking Suite</i> measures performance levels and their stability for different type of workloads in Cloud and Edge infrastructures.</li> <li>▶ <i>T&amp;R Engine</i> aids the measurability of the QoS of Cloud/Edge services through assigning to them and monitoring Trust &amp; Reputation indexes, based on their past performance and reliability. That way, the ranking of the IaaS providers for use in more critical applications can be enabled, based on the trustworthiness of the corresponding services.</li> </ul>
<p>#4 - Define and apply sets of metrics that are meaningful to the end users for QoE and QoS Levels.</p>	<ul style="list-style-type: none"> <li>▶ SLA Lite creates, manages and assesses the SLAs defined in the ConfService UI application.</li> <li>▶ The pilots defined metrics meaningful to determine and improve the QoS and QoE for the end-users. They were defined and specified for the different application areas. <ul style="list-style-type: none"> <li>- UC1 uses file loading time, resolution and bandwidth, which are tied to QoS, but have a high influence on the QoE in streaming media content. Users might experience discomforts in case resolution rises above specified values.</li> <li>- UC2 uses availability to assess the QoS of the service and measures and improves the end-to-end delay to warn the users on time and reduce the risk for accidents.</li> <li>- In the UC3 an analytical service determines the further procedure of the metal-cutting machine. Therefore, the availability of the service is used to determine the QoS. The calculation time for the analytical result results in the availability of the analytical result, which is monitored and improved to ensure that end users do not experience any interruptions in the manufacturing process.</li> </ul> </li> </ul>
<p>#5 - Define and implement new ways of increasing security and privacy on edge, filtering out (during the cloud moving process) with intelligent way the data that have to be kept on the edge.</p>	<ul style="list-style-type: none"> <li>▶ <i>DLT</i> allows by design the endorsement of Privacy Enhancing Technologies (PETs) in its trustless environment adding hard privacy requirements and eliminating third-parties from the architecture transaction level. <i>DLT</i> offers a lightweight blockchain ledger and its corresponding trust-ensuring mechanisms enabling machine-to-machine and human-to-machine transactions with transparency, proper authentication, and authorisation.</li> <li>▶ <i>Whisper protocol</i> empowers end-to-end security of the information flow from the device to the edge and cloud computing over public multipurpose infrastructures by establishing privacy and confidentiality.</li> </ul>
<p>#6 - Pilot and demonstrate the applicability and scalability of the tools on large scale edge/cloud infrastructures and for applications that typically need specialised oversized infrastructures.</p>	<ul style="list-style-type: none"> <li>▶ <i>DSS</i> smart orchestration algorithms support the scaling and edge-cloud-edge offloading to showcase the optimisation of resource consumption, latency, and energy consumption on the edge.</li> <li>▶ <i>E2CO</i> manages the connection to the different infrastructures to realise the actions decided by the DSS (i.e. deployment, migration, horizontal and vertical scaling of applications).</li> </ul>

Objective	Outcome
	<ul style="list-style-type: none"> <li>▶ The <i>SOE</i> and <i>RAN Controller</i> can interact with diverse types of underlying infrastructures, including Kubernetes. By communicating directly with the Kubernetes API server, the management of computational resources is partially delegated to Kubernetes, which supports up to 5,000 nodes in its current release<sup>3</sup>. The <i>SOE</i> and <i>RAN Controller</i> require the allocation of a Kubernetes namespace with a certain resource quota to a given network slice; then, the Kubernetes master components handle the management of resources within that namespace.</li> <li>▶ <i>Benchmarking Suite</i> is designed to run multiple tests on different infrastructures in parallel and (when it does not affect the performance results) run multiple tests on different nodes of the same infrastructure. This allows to assess large infrastructures in an optimised way.</li> <li>▶ <i>App Profilers</i>’ light weight nature as well as the leverage of Node-RED framework allows it to scale and be able to profile multiple containers.</li> </ul>
#7 - Enable the extension of the Edge/Cloud combined business model and provide an extensive replicability and best practices framework.	<ul style="list-style-type: none"> <li>▶ For the best practices, <i>D6.3</i> includes lesson learned by the use cases to highlight the benefits of adopting Pledger as a platform for cloud-edge secure orchestration and monitoring.</li> <li>▶ For the replicability, references are provided on public source repository with detailed instructions about Pledger components installation, configuration and validation of their main features.</li> </ul>

As shown above, all the functional components of Pledger actively participate in the fulfilment of 1 or more of the objectives. More specifically for every individual objective, the outcomes can be encapsulated as follows:

- ▶ **Enhance edge/fog computing provider resource management (objective 1):** Pledger created components that monitor and evaluate various aspects of the provider's resources and services using *Benchmarking Suite* and *SLA Lite* to achieve this goal. Pledger also enables the orchestration and the configuration of the deployment of the resources and sliced networks to govern the QoS and QoE using *DSS* and *SOE and RAN Controller*. Also, *App Profiler* aids the management procedure by delivering information on the computation needs of the deployed applications.
- ▶ **Coupling of edge/fog computing with Blockchains and distributed ledger technologies (objective 2):** *DLT* directly addresses this objective through the development and implementation of a trusted distributed ledger network that is able to operate securely in edge/fog infrastructures. Also, *SLASC* leverages the *DLT* to create an automated coupling of SLA contractual commitments with smart contracts.
- ▶ **Enhance Cloud/Edge services behaviour measurability and predictability (objective 3):** In the context of Pledger, specific components were created in order to assess the behaviour of the services, both from the side of IaaS with the usage of *Benchmarking Suite* and from the side of service adopters with the usage of *App Profiler* to assess the resource requirements of applications or services. It is also worth noting that the *T&R Engine*, through the use of Trust & Reputation indexes, aids in the measurability of the QoS of cloud and edge services.
- ▶ **Define and apply sets of metrics that are meaningful to the end users for QoE and QoS Levels (objective 4):** With the usage of *SLA Lite* users were able to define specific metrics in the form of

<sup>3</sup> [Considerations for large clusters | Kubernetes](#)

SLA parameters that can affect the QoS and QoE. Pledger *Use Cases* actively aided this process, by defining and testing all the important metrics that effected their application or service QoE.

- ▶ **Define and implement new ways of increasing security and privacy on edge (objective 5):** Implementation of *DLT* constitutes one of the main developments on the process of increasing security and privacy, by enabling the enforcement of Privacy Enhancing Technologies and implementation of a trusted Blockchain network. Also *Whisper protocol* enables the secure flow of information in this trusted environment, adding another layer of security on top of *DLT*.
- ▶ **Pilot and demonstrate the applicability and scalability of the tools on large scale edge/cloud infrastructures (objective 6):** All the components responsible for the orchestration and scaling actuation (*E2CO, DSS, SOE and RAN Controller*) were tested in mixed cloud/edge environments using as applicability validator the Use Case applications. As for the scaling of the *App Profiler* and *Benchmarking suite*, both of these components can scale efficiently to facilitate a large number of requests.
- ▶ **Enable the extension of the Edge/Cloud combined business model and provide an extensive replicability and best practices framework (objective 7):** For best practises, section D6.3 highlights the advantages of using Pledger as a platform. For the purpose of reproducibility, references to a public source repository with complete installation, setup, and validation instructions for Pledger components are supplied.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	50 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

## Conclusions

---

In this deliverable, the work performed in the Task 5.4 “Validation and overall evaluation” was described. In this task, the overall evaluation of Pledger was performed in four steps: Validation of the requirements, evaluation of the individual components, evaluation through UCs, and evaluation of objectives.

The requirements validation showed 91.8% successfully and 5.7% partially implemented or slightly changed requirements. During the pilots operation and evaluation, no restrictions or impairments based on requirements which were not satisfied or only implemented were reported. Therefore, the system can be considered as correct and complete.

The individual components were assessed in terms of accuracy, consistency, and improvements of automatization over manual procedures. The improvements for automatising compute and network slice reservation as well as automation of service orchestration were clearly demonstrated, i.e. in the range of a few minutes (manual) vs less than one minute (automated).

Moreover, the Pledger UCs were validated and evaluated. All three UCs demonstrated and evaluated the pilot in the relevant environment. All of them benefitted through the automated service orchestration, automatised performance optimisation based on decision-making and management of sensitive information. End users were involved, and their feedback was gathered. The end users experience was very good with a satisfaction of 93%. Suggestions for further improvements of the QoE were made, like adaptations to alarming signals or adjusting colours or information on UIs.

Overall, all seven objectives defined at the beginning of the project were fulfilled. This was validated by mapping all achieved outcomes to the high-level objectives. Every functional component of Pledger participates in the fulfilment of at least one objective. Furthermore, the UCs contributed to the outcomes both actively by defining relevant metrics for the end users and indirectly by supporting Pledger components in the evaluation to achieve the described outcomes.

All KPIs were reached during the project or were even outperformed, e.g. the experienced performance and stability of virtualised resources through enhanced provider resource management could be improved by 48% compared to the target of 30%.

Lastly, the Pledger system achieved high levels of technical and integration maturity through the corresponding demonstrations and pilots and shows the high professional research and development work and excellent cooperation performed by the partners throughout the project life cycle.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	51 of 69		
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0	<b>Status:</b>	Final

## References

- [1] V. Stanzl “D5.7 Pilots operation and monitoring III”, [Online]. Available: <http://pledger-project.eu/content/deliverables>. Accessed 20 November 2022.
- [2] F. Iadanza and G. Giammatteo, "D2.2 Pledger Requirements Analysis," [Online]. Available: <http://www.Pledger-project.eu/content/deliverables>. Accessed 16 November 2022.
- [3] “MOSCOW Priorisation”, [Online]. Available: [https://www.agilebusiness.org/page/Project-Framework\\_10\\_MoSCoWPrioritisation](https://www.agilebusiness.org/page/Project-Framework_10_MoSCoWPrioritisation). [Accessed 24 November 2022].
- [4] F. Iadanza, “D4.6 Decision Support tools II”, [Online]. Available: <http://pledger-project.eu/content/deliverables>. Accessed 25 November 2022.
- [5] V. Stanzl, “D5.3 Pilots operation and monitoring I”, [Online]. Available: <http://pledger-project.eu/content/deliverables>. Accessed 20 November 2022.
- [6] O. Voutyras, “D2.3 Pledger overall architecture”, [Online]. Available: <http://pledger-project.eu/content/deliverables>. Accessed 24 November 2022.
- [7] “Docker”, [Online]. Available: <https://www.docker.com/>. Accessed 29 November 2022.
- [8] “Kubernetes”, [Online]. Available: <https://kubernetes.io/>. Accessed 29 November 2022.
- [9] “MicroK8s”, [Online]. Available: <https://microk8s.io/>. Accessed 29 November 2022.
- [10] “K3s”, [Online]. Available: <https://k3s.io/>. Accessed 29 November 2022.
- [11] “System Readiness Level Index”, [Online]:[http://wiki.doing-projects.org/index.php/System\\_Readiness\\_Level\\_Index](http://wiki.doing-projects.org/index.php/System_Readiness_Level_Index). Accessed 29 November 2022.
- [12] F. G. Mármol and G. M. Pérez, “Trmsim-wsn, trust and reputation models simulator for wireless sensor networks”, 2009 IEEE International Conference on Communications, pp. 915–919, Piscataway, NJ, USA, 2009.

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation			<b>Page:</b>	52 of 69
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final

## Annex

---

In the following, the list of requirements, their final implementation status, and a reference to the validation of this requirement is given, e.g., where it is demonstrated or described in details. Table 23 shows this validation for the initial functional requirements, and

<b>Document name:</b>	D5.8 Pledger overall validation and evaluation				<b>Page:</b>	53 of 69	
<b>Reference:</b>	D5.8	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0	<b>Status:</b>	Final

Table 24 for the initial non-functional requirements. In M24 of the project, the requirements were updated, and some additional ones were added.

Table 25: List of additional functional requirements and their validation. Table 25 shows the additional functional requirements and their validation, while Table 26 provides the corresponding information for the additional non-functional ones.

**Table 23: List of initial functional requirements and their validation.**

Code	Description	Status/Validation
FR.01	The Pledger system must gather metrics from infrastructure and services, including response time, latency and other metrics collected from the monitoring tools.	<b>COMPLETED</b> The Monitoring Engine gathers these metrics from the different infrastructures (i.e., the Prometheus instances deployed in the testbeds).
FR.02	The Pledger system should provide advice about the best infrastructure for a specific service based on profiling and classification processes for infrastructure and services (automatic or manual).	<b>COMPLETED</b> The Recommender takes the accumulated data from the Benchmarking Suite and the App Profiler, and makes the deployment decision. This has been demonstrated and described in D5.6 Section 3.2.
FR.03	The Pledger system should allow the update or rollback of a new service with little or no interruption for the users (new version, bug fixed, etc.).	<b>COMPLETED</b> New releases are deployed through Kubernetes with rollout update and internal load balancer, so with no disruption.
FR.04	The Pledger system could schedule automatic scaling of services based on SLA violation as far as the tenant has available resources.	<b>COMPLETED</b> The DSS processes the SLA violations and takes the required actions by calling the Orchestrator component, which is the one responsible for scaling the services
FR.05	The Pledger system should allow to remove or update part of the offering to service provider based on infrastructure catalog: new hardware, decommissions, etc.	<b>COMPLETED</b> This is done via the ConfService application.
FR.06	The Pledger system could show historical performance about services and infrastructure.	<b>COMPLETED</b> The SLA subsystem offers a REST API to get information about historical data for SLAs related to apps and infrastructures.
FR.07	The Pledger system could allow the definition of the execution plan of services: max/min instances, schedules.	<b>COMPLETED</b> This is defined in the ConfService tool.
FR.08	The Pledger system could deploy event-driven apps with zero instances (FaaS).	<b>DROPPED</b> No FaaS is needed. Effort was moved to other activities.
FR.09	The Pledger system could implement the following recovery actions after an SLA breach: apps scalability, apps migration to a Cloud or Edge.	<b>COMPLETED</b> The DSS / Recommender process the SLA violations and takes the required actions (i.e. migration, vertical and horizontal scaling, termination) by calling the Orchestrator component, which is the one responsible for scaling the services

Code	Description	Status/Validation
FR.10	The Pledger system could allow to choose in which region, geolocation and specific subset of resources (e.g., availability zones) the data can be uploaded from edge to cloud.	<b>COMPLETED</b> This is defined in the ConfService tool.
FR.11	The Pledger system must allow dynamic configurations before and after service deployment.	<b>COMPLETED</b> The ConfService offers the possibility to adapt settings after the service deployment. Demonstrated in the DSS demos published on Pledger YouTube channel, described in D4.6.
FR.12	The Pledger system must visualise performance statistics about each service in a dashboard.	<b>COMPLETED</b> Using Grafana dashboards, the metrics in Monitoring Engine can be visualised, as demonstrated in UC3 and described in D5.4, Section 6.3.
FR.13	The Pledger system must allow to manage service lifecycle (e.g., starting/stopping) via a dashboard.	<b>COMPLETED</b> Using ConfService, apps can be started and stopped as demonstrated and documented in D5.6, Section 3.1.
FR.14	The Pledger system should be able to share selected datastreams to third-party service providers.	<b>COMPLETED</b> Using StreamHandler and its authorisation mechanisms, specific and selected streams can be shared with third-party providers.
FR.15	The Pledger system must trace data usage/transfer to enable a business model based on pay-per-use approach.	<b>COMPLETED</b> Analysis results are stored on the DLT to trace the parts produced by the machine. This can be accessed only by authorised parties and enables a business model based on pay-per-use of the machine, i.e. parts produced. This is documented in Section 5.3.3.
FR.16	The Pledger system must provide a dashboard to check which data has been released to whom.	<b>DROPPED</b> The release of data is handled via the StreamHandler and a configuration is set about which data is pushed to the StreamHandler.
FR.17	The Pledger system must provide an API to share selected datastreams to third parties.	<b>COMPLETED</b> Only authorised third parties with a valid certificate can consume messages from the topics on StreamHandler as described in D5.2, Section 3.1.2.
FR.18	The Pledger system must be able to offload computation to remote rendering services, in order to overcome the device limitations and ensure functionality.	<b>COMPLETED</b> Offloading scenarios for rendering were changed to a scaling scenario. This is demonstrated in the UC1 and documented in D5.4, Section 4.3.1.
FR.19	The Pledger system should be able to configure the networking between apps deployed on the cloud, the edge, and the end-user devices.	<b>COMPLETED</b> The networking between the apps in different points is enabled natively by Kubernetes, when deploying the containers that contain the apps within one specific namespace. The requirement is that the infrastructure underneath supports the connectivity natively (UC1, UC3) or that it is configured as part of the service creation on top of a slice when using the SOE framework (UC2).

Code	Description	Status/Validation
FR.20	The Pledger system should be maintained by a CI/CD devops development pipeline where bug fixes and features are automatically deployed frequently without disruption.	<b>COMPLETED</b> This is the common methodology applied for the development of the different Pledger components as described in D5.2.
FR.21	The Pledger system must be able to register cloud, edge, and radio infrastructures resources while reflecting the available capabilities (e.g. GPU, etc.) of the registered infrastructure.	<b>COMPLETED</b> Radio and compute resources can be registered. On the radio side V2X (IEEE 802.11p) and 5G NR have been validated and showcased. The UC2 demos (final validations of integration work) and the 5G PoC demonstrate these capacities.
FR.22	The Pledger system must be able to configure and use the computation and radio resources of the citywide testbed that the infrastructure provider makes available.	<b>COMPLETED</b> On the radio side, V2X (IEEE 802.11p) and 5G NR have been validated and showcased. Both the configuration of the radio and the compute resources to later enable slice deployment are possible. The UC2 demos (final validations of integration work) and the 5G PoC demonstrate these capacities together with FR23.
FR.23	The Pledger system should allow to dedicate a part of the testbed-wide resources to a specific tenant, i.e., offer a dedicated slice for computation and radio resources.	<b>COMPLETED</b> The slice creation process has been implemented and fully integrated with the rest of Pledger components. The UC2 demos (final validations of integration work) and the 5G PoC demonstrate these capacities together with FR22.
FR.24	The Pledger system must support multiple connectors to stream big data.	<b>COMPLETED</b> This is achieved through the Kafka-Connect framework which is integrated in the Pledger StreamHandler platform, on top of the Producer and Consumer APIs of Apache Kafka, as explained in D5.2, section 3.1.2.1. With Kafka-connect data streaming can be performed between Apache Kafka and external data systems (e.g., databases) in a high-performance, reusable, and reliable manner. In the context of Pledger, a Kafka connect cluster with three nodes has been configured.
FR.25	The Pledger system should deploy workload in edge and cloud infrastructure.	<b>COMPLETED</b> The Edge-To-Cloud Orchestrator can deploy in Edge and Cloud devices, and can also migrate services from Cloud to Edge and from Edge to Cloud.
FR.26	The Pledger system must import metrics from external sources (monitoring tools, system logs, etc.).	<b>COMPLETED</b> The Monitoring Engine gets the metrics from different Prometheus instances, and also via Kafka messages
FR.27	The Pledger system should allow the configuration of different notification channels (webhook, external API call, email, etc.).	<b>PARTIALLY COMPLETED/CHANGED</b> Notifications are achieved via Kafka and REST API.
FR.28	The Pledger system could support the optimal selection of the least-cost path towards each edge location.	<b>COMPLETED</b> The DSS implements several optimisation algorithms to optimise end-to-end latency, resource usage and energy consumption. Demonstrated with ECODA, TTODA and

Code	Description	Status/Validation
		EA-ECODA algorithms, submitted/published on IEEE papers and described in D4.6
FR.29	The Pledger system could support deployment of apps on edge node with ML capabilities.	<b>PARTIALLY COMPLETED/CHANGED</b> This requirement has been modified, as not enough data was available for ML. The effort has been moved to design and implement Lagrangian optimisations. Demonstrated with DSS demos about ECODA, TTODA, and EA-ECODA algorithms, submitted/published on IEEE papers and described in D4.6
FR.30	The Pledger system must be able to load 3D CAD files which can be viewed live in a room through a HoloLens or a similar device.	<b>COMPLETED</b> The PledgAR Workspace allows users to load CAD files of any extension type and have them floating in front of them.
FR.31	The Pledger system could support blockchain to secure the Signalling (handshake) process between the ISAR Server and the ISAR Client.	<b>COMPLETED</b> The Signalling between the ISAR Client running on the HoloLens and PledgAR Workspace on the edge device is done through the Whisper protocol running on the DLT.
FR.32	The Pledger system must support world anchors or image tracking as reference points, so that AR devices can remember and load the CAD file at the same location for the next iteration.	<b>COMPLETED</b> PledgAR Workspace supports world anchors.
FR.33	The Pledger system must allow collaboration among users through shared experience functionality.	<b>COMPLETED</b> PledgAR Workspace supports the multi-user functionality and can have multiple users in the same session enjoying a shared experience.
FR.34	The Pledger system must allow, during the shared experience sessions, all off-site members to be displayed as avatars to allow a more intuitive interaction.	<b>COMPLETED</b> The users of the multi-user functionality appear as holographic avatars within the session.
FR.35	The Pledger system must allow, when having more than one CAD file loaded, or when displaying a file 1x1 in a real life location, areas of overlapping or collision to be highlighted in red.	<b>COMPLETED</b> PledgAR Workspace has a collision detection system displaying collision in three different manners.
FR.36	The Pledger system must allow the addition of annotations, notes, etc., as well as measuring distances or drawing of basic shapes to the CAD file.	<b>COMPLETED</b> PledgAR Workspace provides annotations such as distance and angle measurements.
FR.37	The Pledger system should export CAD files changes to CAD repositories, so that development improvements are saved and can be	<b>COMPLETED</b> PledgAR Workspace allows you to save the progress made on CAD files so that end-users are able to pick-up where they left off last session.

Code	Description	Status/Validation
	used for the next iteration of the design process.	
FR.38	The Pledger system must allow user end devices to be registered on the services that implement the VRU safety features.	<b>PARTIALLY COMPLETED/CHANGED</b> The application running within the Pledger ecosystem will expose an interface that allows to add/delete specific users (VRUs). Only those users will be able to use the application.
FR.39	The Pledger system must allow the users to be warned by an audiovisual signal in case a risk is detected.	<b>COMPLETED</b> A Gadget was developed and integrated that sends out audio-visual warnings to drivers of a VRU (scooter/bike) if a risk is detected.
FR.40	The Pledger system must allow users to share their position with the system.	<b>COMPLETED</b> This was already tested on-street and shown in the 1 <sup>st</sup> interim review demo, the positions are shared with the UC2 application and from there they can be further processed.
FR.41	The Pledger system must provide dashboards/GUIs to manage the resources/services assigned to the tenant.	<b>COMPLETED</b> Using ConfService, all available infrastructure and services assigned to the tenant can be seen and managed by the user. This is demonstrated in several videos about ConfService available on YouTube, e.g. “review #1 – Pledger ConfService #1”.
FR.42	The Pledger system must be able to process location and sensor inputs to determine risky situations.	<b>COMPLETED</b> This was already tested on-street and shown in the 1 <sup>st</sup> interim review demo.
FR.43	The Pledger system should be able to detect the urban transport (TRAM) presence by reading its shared location or by detecting it via sensors to identify potential risk situations.	<b>COMPLETED</b> The tram location is shared via dedicated on-board unit (OBU) that connects to the risk detection software. Together with the information from the scooters that also use OBUs, the system can detect risky situations. This is being showcased in the pilot experiments.
FR.44	The Pledger system should allow the configuration of the SLA offering by the infrastructure provider.	<b>COMPLETED</b> SLAs are defined using the ConfService application. This includes the guarantees, the SLAs, and the applications linked to these SLAs.
FR.45	The Pledger system should have an SLA template repository.	<b>COMPLETED</b> The SLA subsystem offers the option to add and use SLA templates.
FR.46	The Pledger system could show SLAs details and historical status of SLA violations or fulfilment.	<b>COMPLETED</b> The SLA subsystem offers a REST API to get information about historical data of SLA violations, including the metrics values, the date, the application and infrastructure that produced the violation, etc.
FR.47	The Pledger system should allow the cancelation of SLAs because of some changes, like end of product support, product improvements, discovery of infrastructure incompatibility, etc.	<b>COMPLETED</b> SLAs can be terminated and removed.

Code	Description	Status/Validation
FR.48	The Pledger system could allow the automatic renewal of SLAs in case SLA has begin and end date.	<b>COMPLETED</b> SLAs can be updated and renewed via the SLA REST API.
FR.49	The Pledger system must send notifications regarding SLA violations.	<b>COMPLETED</b> The SLA subsystem uses Kafka and the REST API to notify other Pledger components about SLA violations
FR.50	The Pledger system must allow the sign up of smart contracts of the SLA agreements.	<b>COMPLETED</b> Notifications about new SLAs are sent to Blockchain Subsystem (SLAC Bridge) which is responsible for this task
FR.51	The Pledger system must allow the check of smart contract transactions and violations.	<b>COMPLETED</b> Notifications about SLA violations are sent to Blockchain Subsystem (SLAC Bridge) which is responsible of this task.
FR.52	The Pledger system should be able to provide a trustful network for edge nodes in the context of a permissioned blockchain network where each participant can have enhanced guarantee that the co-participant nodes operate under legal and well-intentioned motives.	<b>COMPLETED</b> The Pledger DLT offers a trustful permissioned blockchain for edge nodes where each participant actor is legally admitted and connected in the network.
FR.53	The Pledger system must be able to execute blockchain network transactions on the edge between participant edge nodes that maintain and are engaged in the blockchain network.	<b>COMPLETED</b> The Pledger DLT is configured with blockchain participant nodes that maintain the network structure and node responsibilities. Blockchain transactions are executed between the participant nodes of the network regardless of the level of privilege and without compromising permission rules. Particularly, the dedicated environments' transactions are holistically validated by the participant nodes of the network.
FR.54	The Pledger system should be able to support node communication in privileged networks that are able to exchange important value inside the Pledger blockchain network.	<b>COMPLETED</b> The Pledger DLT is enabling the architectural deployment of privileged environments that allow for private access to authorised and trusted parties. According to the information logic included in such deployments, sensitive and isolated data are accessed by the corresponding entities.
FR.55	The Pledger system should allow the initiation of smart contract deployment from a user-friendly interface, hiding unnecessary details to the developers.	<b>COMPLETED</b> The Pledger core system allows for the initiation of smart contract deployment through a user-friendly interface. A user is creating an SLA on the ConfService UI, and then SLASC Bridge component automates the smart contract creation and deployment of the blockchain agreement terms on the DLT. Unnecessary details of smart contract execution are hidden to the user.
FR.56	The Pledger system must have user-friendly UI to interact with data stored in the ledger.	<b>COMPLETED</b> The Pledger DLT is providing user-friendly ways to interact with the data stored on the ledger. The deployed user wallets allow for human interaction with the dedicated data submitted, while each supported scenario allows the user for different kinds of access and interaction. For instance,

Code	Description	Status/Validation
		as far as the wallets connected with the SLA configurations are concerned, the dedicated user interaction includes permissions such as check of user balance.
FR.57	The Pledger system must support crypto-token transactions (i.e., Pledger token).	<b>COMPLETED</b> The SLASC Bridge component endorses compensation mechanisms on the DLT allowing for dedicated token transactions. When an SLA violation occurs, the token balances of the SLA parties are changed accordingly to the violation severity through the support of such transactions.
FR.58	The Pledger system should support private smart contracts execution.	<b>COMPLETED</b> The Pledger DLT allows for private smart contract execution inside the isolated private environments. Each DLT activity executes inside the dedicated environment offered, i.e., the SLA to Smart Contract Process along with the SLA Violations Handling are executed privately inside the dedicated DLT environment.
FR.59	The Pledger system must provide a tamper-proof and secure ledger to track the exchange of data among edge nodes.	<b>COMPLETED</b> The Pledger DLT is providing secure and tamper-proof blockchain operations with dedicated ledger privacy. The exchange of data is implemented through the System Integrator that enables the interoperability of the network with authorised modules and components.
FR.60	The Pledger system must detect and identify anomalies, prioritise risks and provide prompt alerts about abnormal security behaviours.	<b>COMPLETED</b> A distributed cybersecurity network streaming platform has been implemented, aiming to offer real-time intrusion detection (IDS) and network security monitoring. The platform is presented analytically in D4.4, sections 4.3.1 and 5.1, and demonstrated in the IDS demo on Pledger YouTube channel.
FR.61	The Pledger system could provide an infrastructure test suite in order to measure and gather performance metrics to classify the infrastructure.	<b>COMPLETED</b> The Pledger System collects performance metrics through the Benchmarking Suite components and uses the collected metrics to classify and rate infrastructure on the basis of their performance.
FR.62	The Pledger system must provide a test suite for workload (applications) in order to measure and gather metrics of performance to classify the services.	<b>COMPLETED</b> This is the very functionality of the Application Profiler developed by ICCS.
FR.63	The Pledger system must integrate reports about benchmarking in a dashboard (e.g., charts).	<b>COMPLETED</b> A visualisation tool based on Grafana is provided with multiple dashboards and charts to visualise and analyse results using multiple filters and aggregations. In addition, ConfService collects and visualises the benchmarking reports for all infrastructures and nodes as a list sortable on different fields.

Code	Description	Status/Validation
FR.64	The Pledger system could schedule the execution of benchmarking tests on an infrastructure with a custom frequency.	<b>COMPLETED</b> The Benchmarking Suite GUI allows to customise the frequency with which execute each schedule. The GUI also allows to specify a time window to schedule the executions. The default execution frequency is every 3 days.
FR.65	The Pledger system could configure private execution on private infrastructure and/or with private benchmarking tests.	<b>COMPLETED</b> The multitenancy model of the Benchmarking Suite allows users to keep their configurations for infrastructures, tests definition, and schedules private or visible to a group of users.
FR.66	The Pledger system should allow users to keep benchmarking results private.	<b>COMPLETED</b> The multitenancy model of the Benchmarking Suite allows users to keep the results for their executions private or visible to a group of users. The users can also set the visibility of benchmarking results in the ConfService.
FR.67	The Pledger system could provide a library of general-purpose micro- and application-level benchmarking tests.	<b>COMPLETED</b> Multiple benchmarking tests have been added.
FR.68	The Pledger system must provide an interface to query benchmarking results.	<b>COMPLETED</b> The Benchmarking Suite provides an API compatible with the InfluxDB query language to query the benchmarking results database. The API requires authentication and returns only results visible to the requesting user.
FR.69	The Pledger system must filter, aggregate, and group benchmarking results.	<b>COMPLETED</b> The Benchmarking Suite allows users to filter and aggregate results in its visualisation tool as well as in the API query endpoint.
FR.70	The Pledger system could provide customizable dashboards with benchmarking results.	<b>COMPLETED</b> The Benchmarking Suite visualisation tool is based on Grafana and it is configured to allow some degree of customisation for charts and dashboard. However, provided dashboards have already several filtering and aggregation capabilities.
FR.71	The Pledger system could execute a benchmarking test provided by the user.	<b>COMPLETED</b> The Benchmarking Suites provides a syntax to define a test, and describe the execution steps and the parsing of results. New tests can be managed in the GUI.
FR.72	The Pledger system must provide a dashboard to highlight the different infrastructure nodes and their availability in terms of computation and network resources quotas.	<b>COMPLETED</b> The ConfService allows configuring different infrastructures and shows capacity/availability of node resources. Demonstrated in the integration demo#1 published on Pledger YouTube channel, and in the ConfService snapshots presented in the D4.6 annex about the infrastructure configuration.
FR.73	The Pledger system must provide a dashboard to highlight the different infrastructure nodes and their usage in terms of computation and network.	<b>COMPLETED</b> The ConfService allows configuring different infrastructures and shows capacity/availability of node resources. Demonstrated in the integration demo#1 published on

Code	Description	Status/Validation
		Pledger YouTube channel, and in the ConfService snapshots presented in the D4.6 annex about the infrastructure configuration.
FR.74	The Pledger system must provide a dashboard to highlight the critical resource allocations.	<b>COMPLETED</b> The DSS monitors SLA violations as feedback for the optimisation algorithms. Demonstrated in the integration demo#1 published on Pledger YouTube channel.
FR.75	The Pledger system must provide a list of possible activity options with respect to different metrics.	<b>COMPLETED</b> The ConfService provides different optimisation algorithms to be configured in the UI. Demonstrated in the DSS demos published on Pledger YouTube channel about the described in D4.6.
FR.76	The Pledger system must provide, for each possible activity option, an efficiency function that weights costs and benefits to help the user pick the best.	<b>COMPLETED</b> The ConfService provides different optimisation algorithms to be configured in the UI. Demonstrated in the DSS demos published on Pledger YouTube channel about the described in D4.6.
FR.77	The Pledger system must allow the user to customise his preferences with respect to different metrics about the best activity option.	<b>COMPLETED</b> The ConfService allows the service provider to specify deployment options priorities and drive the optimisation algorithms. Demonstrated in the DSS demos published on Pledger YouTube channel and shows in the D4.6 annex about the ConfService snapshots.
FR.78	The Pledger system must provide, for each possible activity option, a suggestion about the best choice based on the user preferences, as a notification or a highlighted option in the dashboard.	<b>COMPLETED</b> The ConfService provides different optimisation algorithms to be configured in the UI. Demonstrated in the DSS demos published on Pledger YouTube channel about the described in D4.6. Also, support documentation is provided on Gitlab repository to address the service provider towards the best optimisation for a specific scenario.
FR.79	The Pledger system could support suggestions based on ML algorithms.	<b>PARTIALLY COMPLETED/CHANGED</b> This requirement has been modified, as not enough data were available for ML. The effort was moved to design and implement Lagrangian optimisations. Demonstrated with DSS demos about ECODA, TTODA, and EA-ECODA algorithms, submitted/published on IEEE papers and described in D4.6

Table 24: List of initial non-functional requirements and their validation.

Code	Description	Status/Validation
NFR.01	The Pledger system should expose REST APIs for control/management plane with the HTTPS protocol.	<b>COMPLETED</b> The REST APIs exposed by the SLA and Orchestrator component can be configured to use HTTPS.
NFR.02	The Pledger system should be designed for high availability and fault tolerance of the solution avoiding SPF (single point of failure) as far as possible.	<b>COMPLETED</b> Pledger components are deployed as containerised applications / microservices managed by Kubernetes.
NFR.03	The Pledger system could be tested to support at least a few Cloud Management Toolkits (Kubernetes, OpenStack, etc.)	<b>COMPLETED</b> Pledger supports multiple cloud and edge environments, like Docker, Kubernetes, MicroK8s, and Openshift clusters. Demonstrated in the offloading demo.
NFR.04	The Pledger system should define SLA contracts in JSON format following some standards (predicates, operators, etc.) like JSONPath or similar.	<b>COMPLETED</b> SLAs are defined and stored in JSON format.
NFR.05	The Pledger system could allow infrastructure provisioning (IaC) in declarative format (JSON, YAML, etc.).	<b>PARTIALLY COMPLETED/CHANGED</b> The Edge-to-Cloud Orchestrator and the ConfService applications manage infrastructures defined in JSON format.
NFR.06	The Pledger system must allow to scale-up/down benchmarking executors.	<b>COMPLETED</b> The Benchmarking Suite exploits the scheduling capabilities of Kubernetes and generate new executors on demand to satisfy the request of executions.
NFR.07	The Pledger system could expose a standard querying API for benchmarking results (e.g., PromQL).	<b>COMPLETED</b> The Benchmarking Suite exposes a querying interface that complies with the InfluxQL language defined by Influx DB.
NFR.08	The Pledger system should provide a standard format to include new benchmarking tests (e.g. Docker images).	<b>COMPLETED</b> New tests packaged as Docker Images can be added natively to the Benchmarking Suite that is able to execute them, monitor their execution, and extract metrics.
NFR.09	The Pledger system could have credentials encrypted in the benchmarking backend.	<b>COMPLETED</b> All secrets managed by the Benchmarking Suite are encrypted both in transit (using only https endpoints) and at rest (encrypting the secrets with a password).
NFR.10	The Pledger system could provide an extensible way to exchange metrics and actions for the DSS computation.	<b>COMPLETED</b> Demonstrated in the DSS demos published on Pledger YouTube channel about the different optimisation algorithms available. Described in D4.6.
NFR.11	The Pledger system could support a packaging format that allows the building of new artefacts in few seconds.	<b>COMPLETED</b> Supported by the CI/CD pipeline setup for the project.

Code	Description	Status/Validation
NFR.12	The Pledger system must support a packaging format that allows the instantiation of new artefacts in few seconds.	<b>COMPLETED</b> Demonstrated in the integration demo#1 published on Pledger YouTube channel, and described in D5.6 section 3.1. VMs and containers are supported. The same applies to customisations, based on E2CO.
NFR.13	The Pledger system must support the deployment on edge nodes where apps can run autonomously in case of disconnection from the cloud.	<b>COMPLETED</b> Supported by the different Kubernetes clusters configured for the project.
NFR.14	The Pledger system could support edge orchestrators that are CNCF compatible with Kubernetes.	<b>COMPLETED</b> Demonstrated in the integration demo#1 published on Pledger YouTube channel, and described in D4.6. The DSS can act either as an orchestrator itself, integrating Kubernetes API or send commands to E2CO for enhanced support to different infrastructures.
NFR.15	The Pledger system could support a packaging format with huge availability of artefacts from the open-source community.	<b>COMPLETED</b> Supported by the CI/CD pipeline setup for the project.
NFR.16	The Pledger system must support redundant connection paths for resiliency.	<b>PARTIALLY COMPLETED/CHANGED</b> This is supported in principle, although never tested for lack of redundant network connections in the project.
NFR.17	The Pledger system must support the easy configuration of multi-cloud connections.	<b>COMPLETED</b> Demonstrated in the integration demo#1 published on Pledger YouTube channel, and in the ConfService snapshots presented in the D4.6 annex about the infrastructure configuration.
NFR.18	The Pledger system must support multi tenancy both on cloud and edge resources.	<b>COMPLETED</b> This is natively supported by the different Kubernetes infrastructures configured in the project.
NFR.19	The Pledger system could support connections among large numbers of edge locations having overlapping IP sub-networks.	<b>COMPLETED</b> This is natively supported by the different Kubernetes infrastructures configured in the project.
NFR.20	The Pledger system should integrate with the most popular infrastructures for executing benchmarking tests (e.g., Openstack, Kubernetes, VMWare).	<b>COMPLETED</b> The Benchmarking Suite has connectors for Kubernetes clusters, Docker clusters, VMWare, Openstack, and Amazon AWS. In addition, it has connectors to access any device using SSH protocol.
NFR.21	The Pledger system could support edge nodes that can monitor and restart apps automatically in case of major issues, track and report the failure.	<b>COMPLETED</b> This is natively supported by the different Kubernetes infrastructures configured in the project.
NFR.22	The Pledger system must support a packaging format that supports different GPUs and related SDK (e.g. NVidia).	<b>COMPLETED</b> This is supported in principle, although never tested in terms of CI/CD packaging. UC1 relies on accelerated GPUs, but app images are not packaged through CI/CD.

Code	Description	Status/Validation
NFR.23	The Pledger system must support deployment of apps on edge nodes using low cost HW with possible support to GPUs.	<b>COMPLETED</b> Demonstrated by UC2 using low-cost boards used as far-edge nodes.
NFR.24	The Pledger system could synchronise all data from edge to cloud every 5 seconds.	<b>COMPLETED</b> Broker-To-StreamHandler syncs with even higher frequency in 500ms.
NFR.25	The Pledger system could enable the deployment of applications as Docker containers on edge.	<b>COMPLETED</b> All UC applications are deployed as Docker containers on the edge, as demonstrated in D5.4.
NFR.26	The Pledger system must support training of ML/DL models on the cloud through specialised hardware (e.g. GPU).	<b>DROPPED</b> During the development of the UC applications, the use of ML was disregarded, as the relations in the data were not as complex as initially expected, obviating the need for ML. However, if the training is automatised and packaged as container the use of specialised hardware can be enforced in the deployment options of the service on ConfService.
NFR.27	The Pledger system must enable the transfer of big data from edge to the cloud (15 GB/day).	<b>COMPLETED</b> StreamHandler is capable of transferring big amounts of data and the configuration of the retention period enables the supervision of how long the data is kept to be consumed.
NFR.28	The Pledger system must enable Python to be used as development language for the applications.	<b>COMPLETED</b> The UC3 analytical applications have been developed in Python, as described in D5.1 and D5.5.
NFR.29	The Pledger system must be able to recover from disconnections, work autonomously in the meanwhile and resync data afterwards.	<b>COMPLETED</b> Using the RabbitMQ on edge and the StreamHandler in the cloud, and the publish-subscribe mechanisms enable the buffering of data and results, in case connection is lost. After the connection is re-established, the buffered messages can be consumed.
NFR.30	The Pledger system must transfer data in a secure way to prevent unwanted third-party access.	<b>COMPLETED</b> StreamHandler is used for the data transfer and only authorised parties with a valid certificate can consume messages, as described in D5.2, Section 3.1.2.
NFR.31	The Pledger system must support a packaging format that supports heterogeneous architectures for both cloud and edge, such as X86 and ARM.	<b>COMPLETED</b> Docker and associated docker files are used as packaging format, which enables the support of different architectures, as demonstrated in the offloading scenario, documented in D5.4.
NFR.32	The Pledger system must produce a warning for a VRU needs to be issued in time to avoid any risky situation giving enough time to react (5-10s).	<b>COMPLETED</b> The end-to-end delay must be below 1.5 seconds at all times, according to ETSI TS 101 539-1 requirements for road hazard signalling applications. The DSS provides an upscaling functionality to help achieve this target.
NFR.33	The Pledger system should assure isolation between different services	<b>COMPLETED</b>

Code	Description	Status/Validation
	running in the infrastructure with different tenants.	Isolation among tenants is achieved through network slicing. In Pledger, the SOE and RAN controller provide this functionality.
NFR.34	The Pledger system could require minimum intervention over end user devices.	<b>PARTIALLY COMPLETED/CHANGED</b> In the final design, Pledger does not require an intervention with end user devices. The devices interact only with the risk detection application (service), and not the platform.
NFR.35	The Pledger system must support simple and secure administration for pluggable images (e.g. Docker images).	<b>COMPLETED</b> The Pledger system is currently supporting orchestration environments where images are deployed as containers. The DLT completed an orchestration process throughout the course of Pledger in order to cope with the environment philosophy of the project.
NFR.36	The Pledger system must support decentralised applications which are supported by grouped smart contracts that are programmed to execute autonomously on the blockchain network and deliver different application services when triggered.	<b>COMPLETED</b> The Pledger system supports the triggering of the decentralised applications (DApps) and grouped smart contracts when necessary. For instance, the respective use case DApps execute automatically in case of external triggers and perform the required business intelligence on the ledger.
NFR.37	The Pledger system must be able to ensure data integrity of edge node and user data while keeping specific information on the Pledger blockchain and retrieving the requested data with secure queries.	<b>COMPLETED</b> The Pledger system enables the data integrity and confidentiality on the DLT by design. More specifically, the Pledger DLT is designed with security in mind, while it supports the need for managing sensitive data. Authorised users are able to solely access the data and perform secured queries on the dedicated ledger.
NFR.38	The Pledger system must allow the development of applications running on the Pledger blockchain that enable a modular architecture while providing performance at scale with preserving privacy.	<b>COMPLETED</b> The Pledger system supports modular architecture under the scope of the SLASC Bridge on the DLT. In particular, the DLT is able to process the required transactions at scale, while simultaneously different privacy permissions are allowed on the blockchain network for the respective supported workflows.
NFR.39	The Pledger system must manage big data in compliance with the European legislative framework with respect to data protection.	<b>COMPLETED</b> The big data system offers privacy and security features such as encryption, authorisation, authentication. It is GDPR compliant as well as compliant with the upcoming legislation on Data Governance, as it can operate as a secure data proxy.
NFR.40	The Pledger system must have encryption applied to the big data streams.	<b>COMPLETED</b> This is ensured through the encryption of transmitted data and setup of SSL certificates, making sure that only the sender and the receiver machine can decrypt the traffic that was sent with SSL/TLS. This is described in detail in D5.2, section 3.1.2.2.

Code	Description	Status/Validation
NFR.41	The Pledger system could have authorisation and authentication management for the big data so that only specified connections can access to pub/sub data streams.	<b>COMPLETED</b> Regarding authentication, SSL two-ways authentication and SASL methodologies are available for producers and consumers authentication in the Kafka cluster (StreamHandler). Authorisation is ensured through the availability of Kafka Access Control Lists (ACL) that can limit access to data/topics to specific clients/applications only. The techniques used for authentication and authorisation towards the Kafka cluster are described in detail in D5.2, section 3.1.2.2.
NFR.42	The Pledger system must have secure connections among cluster components and also among communications clients and clusters.	<b>COMPLETED</b> Communication among clusters relies on OpenVPN connections using TLS with client certificates.
NFR.43	The Pledger system must have authenticated access to an easy-to-use interface for the administration of the big data platform.	<b>COMPLETED</b> A Kafka GUI called AKHQ has been set up for the StreamHandler platform administration. AKHQ offers many useful features, including authentication and authorisation. This is described in detail in D5.2, section 3.1.2.1(UI & Monitoring).
NFR.44	The Pledger system must have multiple brokers to maintain replication and load balancing, contributing to the overall elasticity and fault tolerance of Pledger.	<b>COMPLETED</b> Three brokers have been configured within the Kafka cluster of the Pledger platform, to achieve load balancing, replication, and fault-tolerance, as described in D5.2, section 3.1.2.1(Kafka cluster).
NFR.45	The Pledger system must have the capacity to manage data exchanged between core, cloud and edge.	<b>COMPLETED</b> Data transfers to support offloading from Edge to the Cloud is supported through the StreamHandler platform. This is detailed in D5.3, section 3.2.3, covering the requirements of specific Use Cases.
NFR.46	The Pledger system must have the capacity to handle both real time data and batch processing pipelines.	<b>COMPLETED</b> Continuous streams of data are natively supported by the StreamHandler platform. Batch-wise processing is also available where required (UC3), by using a mechanism to synchronise the data and to provide the necessary information on what remains to be processed to the cloud application. This is described in detail in D5.3, section 3.2.3.
NFR.47	The Pledger system should support the capacity to interface with different storages (e.g., DBMS, and so on).	<b>COMPLETED</b> Interfacing with different storage types (DBMS, files, etc.) is performed through the Kafka connect component, integrated in the StreamHandler platform. This is detailed in D5.2, section 3.1.2.1. Moreover, as part of the integration of the StreamHandler platform with the Edge components of UC3, appropriate components have been developed for the data transfers between a database hosted at the Edge and the StreamHandler towards the cloud, as detailed in D5.3, section 3.2.3.

Code	Description	Status/Validation
NFR.48	The Pledger system should enforce security among the edge nodes and among single services at IP level (e.g. with firewalls).	<b>COMPLETED</b> Communication among clusters relies on OpenVPN connections using TLS with client certificates. Within every cluster, communication among nodes (both edge and cloud) is secured by Kubernetes for the control plane. For the data plane, the StreamHandler is securing connectivity among PODs belonging to different clusters.
NFR.49	The Pledger system should provide a security and privacy risk assessment framework to list the trusted nodes that have already proved their ability to comply with the requirements provided by the users.	<b>COMPLETED</b> The Trust and Reputation subsystem is able to assess and monitor the trustworthiness of nodes, in this sense of confidence of the nodes capability (or intention) to provide services of the expected and agreed level. Additional security and privacy assets (IDPS, ML) increase the overall security and trust in the system and filter traffic from trusted sources (i.e. watch for black-listed IPs). For the security assessment, the CIS benchmarks help to identify security issues in the infrastructure nodes and suggested remediation actions according to official best practices from the Center for Internet Security consortium.
NFR.50	The Pledger system must use encryption to enforce storage security.	<b>COMPLETED</b> Encryption at rest is used for all hard disks at the servers (VMs) where the project's artifacts are being deployed. This is detailed in D5.2, section 2.6.2.
NFR.51	The Pledger system must enforce secure communication and authorisation management on the entire CI pipeline.	<b>COMPLETED</b> The connection to the CI/CD services is done over HTTPS to secure the connection of the users to the deployed applications. This boils down to offering three layers of protection: data encryption, data integrity, and users authentication. as detailed in D5.2, section 2.6.1. Authorisation management is achieved by protecting the VMs that host the CI/CD solution with firewalls (iptables), as detailed in D5.2, section 2.6.4.

Table 25: List of additional functional requirements and their validation.

Code	Description	Status/Validation
FR.ADD.1	The Pledger system should allow to configure domain features of the different nodes and infrastructures	<b>COMPLETED</b> Demonstrated in the integration demo#1 published on Pledger YouTube channel, and in the ConfService snapshots presented in the D4.6 annex about the infrastructure configuration.
FR.ADD.2	The Pledger system should support low latency Apps on the edge and allocate them to optimise total latency with respect to resource consumptions	<b>COMPLETED</b> Demonstrated in the DSS demos published on Pledger YouTube channel about the different optimisation algorithms available described in D4.6

Code	Description	Status/Validation
FR.ADD.3	The Pledger system should be able to automatically discover nodes HW features	<b>COMPLETED</b> Demonstrated in the integration demo#1 published on Pledger YouTube channel, and in the ConfService snapshots presented in the D4.6 annex about the infrastructure configuration and the node-feature autodiscovery.
FR.ADD.4	The Pledger system should support vertical scaling of applications on VMWare	<b>COMPLETED</b> Demonstrated in the scaling scenario in D5.4.
FR.ADD.5	The Pledger system should support edge and far-edge devices	<b>COMPLETED</b> Demonstrated in the final integration between Pledger and the SOE Framework. All types of compute resources are available for deploying services: Far-Edge, Edge, Main DC.
FR.ADD.6	The Pledger system should allow to prioritise node/infrastructures and offload accordingly when resources are not available	<b>COMPLETED</b> Demonstrated in the offloading scenario described in D5.4.
FR.ADD.7	The Pledger system should allow to distinguish between SLA which depends on allocated resources (e.g., number of frames processed per second) and those that are not (e.g., failure of radio access network)	<b>COMPLETED</b> Demonstrated in the final integration. DSS can configure SLA to be either used or ignored in the decision process.

Table 26: List of additional non-functional requirements and their validation.

Code	Description	Status/Validation
NFR.ADD.1	The Pledger system should allow the integration with remote infrastructure without need to expose services on the public Internet	<b>COMPLETED</b> A site-to-site OpenVPN configuration was configured with reverse tunnel connection so that OpenVPN servers are publicly exposed only on ENG and i2CAT, as described in D5.4.
NFR.ADD.2	The Pledger system should allow to orchestrate infrastructures using a distributed architecture.	<b>PLANNED</b> Not implemented in the project, gives space for future developments.
NFR.ADD.3	The Pledger system should allow to integrate with monitoring easily and without need to setup dedicated gateway to collect data.	<b>COMPLETED</b> Monitoring application is integrated with the other components via REST API and Apache Kafka. This monitoring system gathers the metrics from the different Prometheus instances deployed in the different testbeds, which are all accessible from the VPNs.
NFR.ADD.4	The Pledger system should allow to assess the security capabilities of the infrastructures to help the SP to prioritise the Nodes where to deploy Apps.	<b>COMPLETED</b> Demonstrated in the CIS security demo published on Pledger YouTube channel about the security scan done on the different infrastructures configured in the project.