D5.2
Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations
21/07/2021
The DECENTER Consortium has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

<table>
<thead>
<tr>
<th>Date</th>
<th>Edited by</th>
<th>Status</th>
<th>Changes made</th>
</tr>
</thead>
<tbody>
<tr>
<td>01.03.2021</td>
<td>Ángel Soriano</td>
<td>Plan</td>
<td>Table of contents (ToC)</td>
</tr>
<tr>
<td>22.04.2021</td>
<td>Maia Buzuleciu</td>
<td>In progress</td>
<td>Update UC1</td>
</tr>
<tr>
<td>22.04.2021</td>
<td>Petar Kochovski</td>
<td>In progress</td>
<td>Work on section 3.3</td>
</tr>
<tr>
<td>23.04.2021</td>
<td>Petar Kochovski</td>
<td>In progress</td>
<td>Work on section 4.3</td>
</tr>
<tr>
<td>27.04.2021</td>
<td>Adrián Arroyo</td>
<td>In progress</td>
<td>Update section 2 and 3</td>
</tr>
<tr>
<td>28.04.2021</td>
<td>Jaewon Moon</td>
<td>In progress</td>
<td>Update UC4</td>
</tr>
<tr>
<td>05.05.2021</td>
<td>Ángel Soriano</td>
<td>In progress</td>
<td>Update Intro and section 4.2</td>
</tr>
<tr>
<td>05.05.2021</td>
<td>Sofia Kleisarchaki</td>
<td>In progress</td>
<td>Update DT sections</td>
</tr>
<tr>
<td></td>
<td>Levent Gurgen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Name</td>
<td>Status</td>
<td>Task</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>06.05.2021</td>
<td>Silvio Cretti</td>
<td>In progress</td>
<td>Work on section 2.1</td>
</tr>
<tr>
<td>07.05.2021</td>
<td>Roi Sucasas</td>
<td>Working</td>
<td>Update section 4.3</td>
</tr>
<tr>
<td>07.05.2021</td>
<td>Raffaele Giaffreda</td>
<td>Working</td>
<td>Work on section 3.1</td>
</tr>
<tr>
<td>11.05.2021</td>
<td>Klaus Kim</td>
<td>Working</td>
<td>Work on section 2.1.2</td>
</tr>
<tr>
<td>18.05.2021</td>
<td>Seungwoo Kum</td>
<td>Working</td>
<td>Work on section 2.2.1 and 2.2.2</td>
</tr>
<tr>
<td>19.05.2021</td>
<td>Ángel Soriano</td>
<td>Working</td>
<td>Integration and update UC2</td>
</tr>
<tr>
<td>20.05.2021</td>
<td>Seungwoo Kum</td>
<td>Working</td>
<td>Work on section 2.2.1 and 2.2.2</td>
</tr>
<tr>
<td>24.05.2021</td>
<td>Soonho Lee</td>
<td>Working</td>
<td>Work on section 2.1.7</td>
</tr>
<tr>
<td>27.05.2021</td>
<td>Petar Kochovski</td>
<td>In progress</td>
<td>Update section 2.2.1 and 2.2.2</td>
</tr>
<tr>
<td>07.06.2021</td>
<td>Raffaele Giaffreda</td>
<td>Working</td>
<td>Update UC1 description and tables</td>
</tr>
<tr>
<td>08.07.2021</td>
<td>Ángel Soriano</td>
<td>Working</td>
<td>Work on section 4.3.1</td>
</tr>
<tr>
<td>11.07.2021</td>
<td>Petar Kochovski</td>
<td>Working</td>
<td>Work on section 4.4.1</td>
</tr>
<tr>
<td>12.07.2021</td>
<td>Raffaele Giaffreda</td>
<td>Working</td>
<td>UC1 update</td>
</tr>
<tr>
<td>15.07.2021</td>
<td>Maia Buzuleciu</td>
<td>Review</td>
<td>Final internal review</td>
</tr>
<tr>
<td>15.07.2021</td>
<td>Soonho Lee</td>
<td>Review</td>
<td>Final internal review</td>
</tr>
<tr>
<td>16.07.2021</td>
<td>Ángel Soriano</td>
<td>Final</td>
<td>Final version</td>
</tr>
</tbody>
</table>

Notice that other documents may supersede this document. A list of latest public DECENTER deliverables can be found at the DECENTER Web page at [https://www.decenter-project.eu/](https://www.decenter-project.eu/).

**Copyright**

This report is © DECENTER Consortium 2018. Its duplication is restricted to the personal use within the Consortium, funding agency and project reviewers.
Acknowledgements

This project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement no. 815141 (DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence)

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT/IITP) (No. 1711075689, Decentralised cloud technologies for edge/IoT integration in support of AI applications).

The partners in the project are FONDAZIONE BRUNO KESSLER (FBK), ATOS (ATOS), KENT, COMUNE DI TRENTO (TN), ROBOTNIK (ROB), UNIVERZA V LJUBLJANI (UL), KOREA ELECTRONICS TECHNOLOGY INSTITUTE (KETI), GLUESYS (GLSYS), DALIWORKS (DW), LG U+ (LGUP), SEOUL NATIONAL UNIVERSITY (SNU).

The content of this document is the result of extensive discussions within the DECENTER © Consortium as a whole.

More information

Public DECENTER reports and other information pertaining to the project are available through DECENTER public Web site under http://www.decenter-project.eu.
# Contents

Revision history ........................................... 1  
Copyright .................................................. 2  
Acknowledgements ........................................ 3  
More information ......................................... 3  
List of figures .............................................. 6  
List of Tables .............................................. 8  
Executive Summary ....................................... 9  

1. Introduction ............................................ 10  
   1.1. Target audience .................................... 10  
   1.2. Related documents ................................ 10  
   1.3. Organization of the document .................... 10  

2. Final Integration of the DECENTER Platform ............ 12  
   2.1. Final DECENTER Platform Services .............. 12  
      2.1.1. Orchestrator .................................. 12  
      2.1.2. Monitoring System ............................ 12  
      2.1.3. SLA Manager .................................. 15  
      2.1.4. L-ADS ........................................ 18  
      2.1.5. Resource Selector ............................. 20  
      2.1.6. Resource Seller ............................... 20  
      2.1.7. IoT Platform .................................. 20  
   2.2. Final DECENTER Application Services, Methods and Tools integration ......... 22  
      2.2.1. AI Package .................................... 22  
      2.2.2. Data Management .............................. 23  

3. Final Integration of Use-case AI Applications .......... 26  
   3.1. Smart City Crossing Safety ....................... 26  
      3.1.1. Integration with DECENTER Platform Services .... 26  
      3.1.2. Integration with DECENTER Application Services, Methods and Tools .... 26  
      3.1.3. Integration of AI Services .................... 27  
   3.2. Robotics Logistics ................................ 28  
      3.2.1. Integration with DECENTER Platform Services .... 28  
      3.2.2. Integration with DECENTER Application Services, Methods and Tools .... 30  
      3.2.3. Integration of AI Services .................... 31  
   3.3. Smart and Safe Construction ....................... 33  
      3.3.1. Integration with DECENTER Platform Services .... 33
3.3.2. Integration with DECENTER Application Services, Methods and Tools
3.3.3. Integration of AI Services

3.4. Ambient Intelligence for Office Environments
  3.4.1. Integration with DECENTER Platform Services
  3.4.2. Integration with DECENTER Application Services, Methods and Tools
  3.4.3. Integration of AI Services

4. Final Demonstrators and pilots
  4.1. Digital Twin KPIs
  4.2. Smart City Crossing Safety
    4.2.1. MVD
    4.2.2. KPIs
      a. DT and SL KPIs
      b. UC1 KPIs
    4.2.3. Final Pilot Description
  4.3. Logistics Robotics Optimization
    4.3.1. MVD
    4.3.2. UC2 KPIs
    4.3.3. Final Pilot Description
  4.4. Smart and Safe Construction
    4.4.1. MVD
    4.4.2. UC3 KPIs
    4.4.3. Final Pilot Description
  4.5. Ambient Intelligence for Office Environments
    4.5.1. MVD
    4.5.2. KPIs
      a. DT and SL KPIs
      b. UC4 KPIs
    4.5.3. Final Pilot Description
  5. Conclusions
  6. Abbreviations
  7. Annex 1: KPI updates
List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User Graphical Interface Log in.</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>Robotnik’s Use Case Flow.</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>Sequence diagram of SLA integration in use case 2.</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Orchestrator of UC4</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>Monitoring System of UC4</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>Dashboard of UC4: the status of deployed containers</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>Dashboard of UC4: monitoring of AI services output and processing time</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>Schema of Digital Twin integration at UC4</td>
<td>37</td>
</tr>
<tr>
<td>9</td>
<td>Flow of UC4 involving Digital Twin and AI Model Repository</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>AI services included at UC4</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>Smart Crossing application components</td>
<td>44</td>
</tr>
<tr>
<td>12</td>
<td>Smart Crossing dashboard, showing identified vehicles/pedestrians, events and system reaction times</td>
<td>51</td>
</tr>
<tr>
<td>13</td>
<td>Advantages of using DECENTER deployment</td>
<td>52</td>
</tr>
<tr>
<td>14</td>
<td>Containerization of real robot modules classification</td>
<td>53</td>
</tr>
<tr>
<td>15</td>
<td>Microservices at DECENTER Fog Platform Interface</td>
<td>55</td>
</tr>
<tr>
<td>16</td>
<td>Rb1 Base Robot from Robotnik company</td>
<td>55</td>
</tr>
<tr>
<td>17</td>
<td>Carts to move material or goods</td>
<td>56</td>
</tr>
<tr>
<td>18</td>
<td>From left to right: person, robot, wooden box</td>
<td>59</td>
</tr>
<tr>
<td>19</td>
<td>Person detection results</td>
<td>59</td>
</tr>
<tr>
<td>20</td>
<td>50 samples of the AI Object recognition service when a robot is in front of it</td>
<td>60</td>
</tr>
<tr>
<td>21</td>
<td>100 tests with person and robot detection cases</td>
<td>61</td>
</tr>
<tr>
<td>22</td>
<td>Robot laser measures shown as red points</td>
<td>62</td>
</tr>
<tr>
<td>23</td>
<td>Measuring manually the distance between the robot and the object</td>
<td>62</td>
</tr>
<tr>
<td>24</td>
<td>Comparison between real distances measured by hand and distances calculated by robot</td>
<td>63</td>
</tr>
<tr>
<td>25</td>
<td>Distance average measurement error</td>
<td>63</td>
</tr>
<tr>
<td>26</td>
<td>Standard deviation error of the distance measures</td>
<td>64</td>
</tr>
<tr>
<td>27</td>
<td>CPU Consumption of involved devices during the vertical offloading test</td>
<td>65</td>
</tr>
<tr>
<td>28</td>
<td>Total CPU usage of the whole cluster during the vertical offloading test</td>
<td>65</td>
</tr>
<tr>
<td>29</td>
<td>CPU consumption during the horizontal offloading</td>
<td>66</td>
</tr>
<tr>
<td>30</td>
<td>Cluster CPU consumption during the horizontal offloading</td>
<td>67</td>
</tr>
<tr>
<td>31</td>
<td>Scenario Representation (left). Real scenario (right)</td>
<td>67</td>
</tr>
<tr>
<td>32</td>
<td>Scenario where there is a box in front of the robot</td>
<td>68</td>
</tr>
<tr>
<td>33</td>
<td>Outcome of AI microservice (left). Picture sent by robot (right)</td>
<td>68</td>
</tr>
<tr>
<td>34</td>
<td>Scenario where a person is in front of the robot</td>
<td>69</td>
</tr>
<tr>
<td>35</td>
<td>“Person” output from AI microservice (left). Picture of person sent by robot (right)</td>
<td>69</td>
</tr>
<tr>
<td>36</td>
<td>Outcome from AI microservice returning “robot” identified (left). Picture sent by robot(right)</td>
<td>70</td>
</tr>
<tr>
<td>37</td>
<td>AI model response time performance (KPI1)</td>
<td>73</td>
</tr>
<tr>
<td>38</td>
<td>Comparison between original and DECENTER mode.</td>
<td>83</td>
</tr>
<tr>
<td>39</td>
<td>Member verifier example</td>
<td>84</td>
</tr>
<tr>
<td>40</td>
<td>Camera used in UC4 to collect input data for the member verification process.</td>
<td>85</td>
</tr>
<tr>
<td>41</td>
<td>Example of person in front of the camera</td>
<td>86</td>
</tr>
<tr>
<td>42</td>
<td>Example of content provided to visitors according to their membership.</td>
<td>86</td>
</tr>
</tbody>
</table>
D5.2 Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations

Figure 43. Example of App composer for UC4 87
Figure 44. Deployment of UC4 by App composer FogAtlas. 87
Figure 45. List of microservices' pods deployed on the DECENTER platform for UC4, showing their status, queried through kubectl Kubernetes command line interface 87
Figure 46. Monitoring system. 88
Figure 47. Model replacement from version 0.1 to version1. 88
List of Tables

Table 1. Services exposed 14
Table 2. AI Services Integrated 26
Table 3. UC2 Application Services (AI Package) Integration 28
Table 4: UC2 AI Service 29
Table 5. AI Services Integrated 33
Table 6. Face_image_dt format 38
Table 7: Member_result format 38
Table 8. Functionalities of AI Applications 39
Table 9. Inputs and outputs of AI services 40
Table 10. Types of connection with other microservices 40
Table 11. Specifications of microservices 41
Table 12. AI Services Integrated 41
Table 13. KPIs of DT and SL 43
Table 14. List of functional requirements for the UC1 45
Table 15. Non-functional Requirements addressed by Y3. 46
Table 16. Results of the DT KPIs. 47
Table 17 Functional Requirements with ontology relationship 48
Table 18. Completeness of DT KPI. 48
Table 19 Relation between Ontologies and UC1 domain 49
Table 20. Conciseness of SL KPI 49
Table 21. Ontologies related with UC1 and UC4 49
Table 22. Adaptability of KPI to UC1 50
Table 23. Ontology metrics description 50
Table 24. Ontology metric results 50
Table 25. FR covered at Y3 57
Table 26. NFR covered by Y3. 58
Table 27. FR covered by Y3. 72
Table 28. NFR covered by Y3. 72
Table 29. Estimation cost of service providers 74
Table 30. KPIs before and after DECENTER 75
Table 31. FR coverage 77
Table 32. NFR coverage 79
Table 33. Results of DT KPIs 79
Table 34. FR contrasted with existing entities in the ontology. 80
Table 35. Completeness of SL. 81
Table 36. List of the ontologies related with UC4 81
Table 37. Conciseness of SL on UC4 81
Table 38. The size and the number of parameters of the original network and the pruned network 83
Table 39. Average precision on WIDER FACE validation dataset 83
Table 40. Experimental results of KPI2 demosntration 84
Table 41. The number of parameters (params) of the models of edge and cloud. 85
Table 42. The member verification accuracy of the models of edge and cloud. 85
Executive Summary

Deliverable D5.2 is the final and updated version of D5.1: First release of the AI-Integrated fog computing platform, demonstration KPIs and first setup of pilots. D5.2 summarizes and updates the work done in tasks T5.1, T5.2 and T5.3 of WP5. The integration work reported in this document shows the final achievements reached and released at M36 as the final results of the integration of DECENTER platform applied to the different use cases within the project.

Like D5.1, this report represents the outcome of the joint effort carried out by European and Korean partners to integrate the DECENTER platform and the use-case AI applications on top of it. The deliverable has been carried out in close collaboration between technical work packages WP3 (in charge of the DECENTER platform services), WP4 (developing the common application services provided by the DECENTER platform), and WP5, where most integration work, especially that aiming to develop and operate use-case specific AI solutions, including demonstration and pilots, was done.

The updates have been presented at three different architecture levels following the scheme of D5.1: DECENTER platform services, DECENTER application services and use-case AI applications.

The final version of demonstration and UC pilots have been described at the last section of the document. This section also includes the explanation of KPIs, its demonstration and its results.
1. Introduction

Deliverable D5.2 summarizes the work done within Work Package 5 (WP5) at M36. The document describes the final updates about the integration work done from the point of view of the following three main perspectives:

- The integration of DECENTER platform as a combination of advanced AI modular solutions provided by DECENTER within WP4 and the DECENTER platform services developed in WP3.
- AI applications integration (investigated and designed in WP4) over the different use-cases into the DECENTER platform.
- The description of the final demonstration of pilots use-case and the analysis and the results of KPIs associated to the different components integrated.

1.1. Target audience

This document is oriented to guide to engineers engaged in the development and operation of AI applications on the DECENTER platform. The integration work applied to the different use cases of the project is explained in detail with the objective to orient and guide the audience to understand the steps needed to apply the DECENTER platform to any other use case.

1.2. Related documents

This deliverable is the final and second iteration of D5.1: First release of the AI-integrated fog computing platform, demonstration KPIs and first setup of pilots, released at M24. This D5.2 relates and refers the following deliverables:

- D3.3. Second release of the fog computing platform (M24)
- D3.4. Final release of the fog computing platform (M30)
- D3.5. Methods and solutions to achieve security and robustness (M36)
- D4.1 First release of applications’ Artificial Intelligence methods and solutions (M12)
- D4.3 Second release of applications’ Artificial Intelligence methods and solutions (M24)
- D4.4 Final report and release on cross-border application data management and use-case specific AI solutions (M30)
- D4.5 Final release of Artificial Intelligence algorithms for distributed applications (M36)
- D5.1 First release of the AI-integrated fog computing platform, demonstration and KPIs and first setup of pilots. (M24)

1.3. Organization of the document

This document has been organized as follows: Section 2 presents the final integration of the DECENTER platform, including platform services (subsection 2.1) and application services,
methods, and tools (subsection 2.2). At section 3, the integration details of AI applications of DECENTER platform have been described focusing on its application in each use case. This third section is divided into four subsections where the details about the integration are explained particularly for each use-case. Section 4 describes the final description of demonstrations and pilot scenarios of the project at M36, including the specification of KPIs, the methodology followed to demonstrate them, and the analysis of the results achieved. Finally, section 5 draws conclusions.
2. Final Integration of the DECENTER Platform

This section explains the last DECENTER components integrated from the delivery of D5.1. Only new components and updates have been described to not duplicate information. Each component details the procedures to install it, its offered API to interact with it and its integration with Kubernetes.

2.1. Final DECENTER Platform Services

The following components have been created or updated during the last year of the DECENTER project.

2.1.1. Orchestrator

During Y3, the Orchestrator component mainly evolved in its internals (e.g., new placement algorithms, support for heterogeneous CPU). Therefore, the installation procedure, the offered API and the integration with Kubernetes remain the same as the ones described in D5.1.

The only relevant external change developed in Y3 is the finalization of the federation of multiple clusters based on the Kubernetes Federation. This new feature, described in D3.5, requires the installation of a new Kubernetes controller managing the federation level orchestration and the setup of Istio\(^1\). For the installation of Istio, see the corresponding documentation while the installation of the federated controller is carried out through helm as for the other Orchestrator components. Relevant files for its deployment can be found in the DECENTER gitlab\(^2\).

A detailed description of each component of the Orchestrator is provided in D3.4.

2.1.2. Monitoring System

The monitoring system can check the status of the overall nodes of the DECENTER cluster and the connection status between microservices.

Integration with other DECENTER components can be found in previous deliverables for monitoring queries and dashboards, D3.3.

Installation

Can check how to install the monitoring system and how to set up the dashboard at GitLab, in the following URL: [https://gitlab.fbk.eu/decenter/w3t1/monitoring-system](https://gitlab.fbk.eu/decenter/w3t1/monitoring-system).

Offered API

The Monitoring System sends a query to the REST API and provides the results for the query.

---

\(^1\) [https://istio.io](https://istio.io)

\(^2\) Accessible only to logged in users: [https://gitlab.fbk.eu/decenter/w3t1/fedfaapp-controller](https://gitlab.fbk.eu/decenter/w3t1/fedfaapp-controller)
D5.2 Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations

It can be obtained by sending a query statement to the end-point /api/v1/query provided of the IP of the monitoring server with a query on the node or microservice of the cluster.

Information about the query at the following URL:
https://gitlab.fbk.eu/decenter/w3t1/monitoring-system/-/tree/master/Use%20Case/UC1/Query_Sample

For example:

Can get the node’s CPU usage in the following way (Example)

- Query = ( 1 - avg(sum by(mode)(irate(node_cpu_seconds_total{mode="idle",instance="$node"}[2m])) / count(count(node_cpu_seconds_total{instance="$node"}) by (cpu)) )* 100

- Monitoring Server IP: 192.168.0.10

  curl http://192.168.0.10/api/v1/query -d query="( 1 - avg(sum by(mode)(irate(node_cpu_seconds_total{mode="idle",instance="$node"}[2m])) / count(count(node_cpu_seconds_total{instance="$node"}) by (cpu)) )* 100"

Integration with Kubernetes

- **Docker image(s)**

  They can be obtained from Docker Hub.

  Prometheus Docker Images

  - node-exporter: https://hub.docker.com/r/prom/node-exporter
  - server: https://hub.docker.com/r/prom/prometheus
  - kube-state-metric: https://hub.docker.com/r/prom/kube-state-metric
  - alertmanager: https://hub.docker.com/r/prom/alertmanager
  - pushgateway: https://hub.docker.com/r/prom/pushgateway

  Grafana Docker Images

  - grafana: https://hub.docker.com/r/grafana/grafana

- **Kubernetes resource files**

  The monitoring server distributes components necessary for monitoring, such as server, pushgateway, node exporter, alertmanager, and dashboard, as configuration files called values yaml files.

  Deployment files can be found in https://github.com/helm/charts.git

  If download “git clone https://github.com/helm/cahrts.git”, you can get the necessary files for distribution.

  The path of the monitoring server configuration file is as follows:
  chart/stable/prometheus/values.yaml

  The configuration file path of the dashboard is as follows.: chart/stable/grafana/values.yaml
Prometheus values.yaml: Configuration file required for distribution of monitoring server, pushgateway, exporter, and alertmanager.

It is distributed as the setting values of the monitoring components through the values file. The following is the server part included in values.

```yaml
server:
  enabled: true  # enable => Whether to install the corresponding component
  name: server
  sidecarContainers:
    image:
      repository: prom/prometheus  # Monitoring server image in docker hub
      tag: v2.20.1
      pullPolicy: IfNotPresent  # If there is a Docker image in the local hub, do not pull the image.
  service:
    annotations: {}
    labels: {}
    clusterIP: ""
    externalIPs: []  # External IP used when you want to access the monitoring server from outside
    loadBalancerIP: ""
    loadBalancerSourceRanges: []
    servicePort: 80
    sessionAffinity: None
    type: ClusterIP
```

Grafana values.yaml: Configuration file for distributing monitoring dashboard

It is distributed as the setting values of the dashboard components through the values file. The following is a part of values.

```yaml
image:
  repository: grafana/grafana
  tag: 7.1.1
  sha: ""
  pullPolicy: IfNotPresent
  securityContext:
```

- **Steps to deploy**
  - Monitoring Install command:
    ```bash
    helm install monitor stable/prometheus -f
    ```
2.1.3. SLA Manager

The SLA Manager is an open-source framework, licensed under Apache Licenses 2.0, responsible for managing service-level agreements between service providers and consumers. Its functionalities and integration with other DECENTER components have been described in previous deliverables, D3.3 and D3.4.

Installation

The SLA Manager and GUI applications are both provided as docker images and their code is available in GitLab, in the following URL: https://gitlab.fbk.eu/decenter/w3t1/sla-management

To install the application from repository code, git and golang are required.

Containerized / Docker images can be downloaded from the following URLs located in Docker Hub:

- SLA Manager: https://hub.docker.com/r/atosdecenter/slalite_dec
- SLA Graphical User Interface: https://hub.docker.com/r/atosdecenter/slalite_dec-ui

These two applications can be installed in Docker or in a Kubernetes cluster. To install both of them in Docker, execute the following command:

```
```

The SLA Manager needs to map port 8090 to be accessed by third party applications. Apart from the ports, the following environment variables need to be defined:

- rebUrl
- rebPath
- brokerURL
- brokerTopic
- adapter
- prometheusUrl

Offered API

The SLA Manager exposes a REST API to third party applications to create and manage SLAs. The services exposed by this API are the following:
<table>
<thead>
<tr>
<th>Method</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/agreements</td>
<td>Returns all agreements</td>
</tr>
<tr>
<td>GET</td>
<td>/agreements/{id}</td>
<td>Gets the information of a specific agreement</td>
</tr>
<tr>
<td>POST</td>
<td>/agreements</td>
<td>Creates a new SLA</td>
</tr>
<tr>
<td>PUT</td>
<td>/agreements/{id}/start</td>
<td>Starts the SLA’s evaluation process</td>
</tr>
<tr>
<td>PUT</td>
<td>/agreements/{id}/stop</td>
<td>Stops the SLA’s evaluation process</td>
</tr>
<tr>
<td>PUT</td>
<td>/agreements/{id}/terminate</td>
<td>Terminates the SLA’s</td>
</tr>
<tr>
<td>PUT</td>
<td>/agreements/{id}</td>
<td>Updates the information of an SLA</td>
</tr>
<tr>
<td>DELETE</td>
<td>/agreements/{id}</td>
<td>Deletes an SLA</td>
</tr>
<tr>
<td>GET</td>
<td>/agreements/{id}/details</td>
<td>Gets all the details of an agreement</td>
</tr>
</tbody>
</table>

Table 1. Services exposed.

Integration with Kubernetes

To deploy these two applications in a Kubernetes cluster the following is required:

- **Docker image(s)**

  They can be obtained from Docker Hub.
  - SLA Manager: [https://hub.docker.com/r/atosdecenter/slalite_dec](https://hub.docker.com/r/atosdecenter/slalite_dec)
  - GUI: [https://hub.docker.com/r/atosdecenter/slalite_dec-ui](https://hub.docker.com/r/atosdecenter/slalite_dec-ui)

- **Kubernetes resource files**

  Deployment files can be found in [https://gitlab.fbk.eu/decenter/w3t1/sla-management/-/tree/master/k8s](https://gitlab.fbk.eu/decenter/w3t1/sla-management/-/tree/master/k8s). There are three files:

  - **slalite-app.yaml**: used to deploy the SLA Manager application and the correspondent service:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: slalite-app
namespace: decenter-tests
labels:
  app: slalite
tier: api
spec:
  replicas: 1
selector:
  matchLabels:
    app: slalite
tier: api
template:
```
D5.2 Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations

```yaml
metadata:
  labels:
    app: slalite
tier: api
spec:
  containers:
    - name: slalite
      image: atosdecenter/slalite_dec:1.0.4
      imagePullPolicy: Always
      ports:
        - containerPort: 8090
---
apiVersion: v1
kind: Service
metadata:
  name: slalite
  namespace: decenter-tests
  labels:
    app: slalite
tier: api
spec:
  selector:
    app: slalite
tier: api
  ports:
    - protocol: TCP
      port: 8090
      targetPort: 8090
      nodePort: 32000
      type: NodePort
    - slalite-ui.yaml needed to deploy the GUI and the service to expose the application:

```
- **kustomization.yaml**: the customization file used to install both applications:

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
  - slalite-app.yaml
  - slalite-ui.yaml
```

- **Steps to deploy**

To deploy the SLA Manager REST API application and the GUI, Kubernetes users have to execute the following command:

```
kubectl apply -f kustomization.yaml
```

2.1.4. L-ADS

The L-ADS (Live Anomaly Detection System) is an asset developed with the aim to detect possible threats in a certain environment. The asset contains two main services: L-ADS and Softflowd. Both are working together to identify possible anomalies on the microservices, the L-ADS is the responsible to classify if the connections captured by the Softflowd are anomalous or legit.

**Installation**

The L-ADS asset has been developed as a docker image. It is possible to find more information related to the asset in the following URL [https://gitlab.fbk.eu/decenter/w3t4/security-of-microservices](https://gitlab.fbk.eu/decenter/w3t4/security-of-microservices) or in the Deliverables D3.3 and D3.5. The communication between the L-ADS and the Softflowd is established in a determinate port. In this case, the L-ADS needs to expose the port 9000.

**Integration with Kubernetes**
To deploy the L-ADS and Softflowd applications in a Kubernetes cluster the following is required:

- **Docker image(s)**

  The containerized version of these two applications can be obtained from Docker Hub:
  - L-ADS: [https://hub.docker.com/r/atosdecenter/lads](https://hub.docker.com/r/atosdecenter/lads)
  - Softflowd: [https://hub.docker.com/r/atosdecenter/softflowd](https://hub.docker.com/r/atosdecenter/softflowd)

- **Kubernetes resource files**

  As both applications are tightly coupled, they have been defined in a single pod (`pods.yaml`):

  ```yaml
  # lads & Softflowd #
  # lads & Softflowd #
  apiVersion: v1
  kind: Pod
  metadata:
    labels:
      io.kompose.service: lads-softflowd
    name: lads-softflowd
    namespace: decenter
  spec:
    containers:
    - image: atosdecenter/lads
      name: lads
      ports:
        - containerPort: 9000
      resources: {}
      stdin: true
      tty: true
    - image: atosdecenter/softflowd:1.0
      name: softflowd
      resources: {}
      tty: true

  ---
  apiVersion: v1
  kind: Service
  metadata:
    labels:
      io.kompose.service: lads-softflowd
    name: lads-softflowd
    namespace: decenter
  spec:
    ports:
    - name: "9000"
      port: 9000
      targetPort: 9000
    selector:
      io.kompose.service: lads-softflowd
  
  ```

- **Steps to deploy**

  To deploy the SLA Manager REST API application and the GUI, Kubernetes users have to execute the following command:
2.1.5. Resource Selector

A detailed description of the Resource Selector component can be found in D3.4.

**Installation**

Resource Selector can be deployed using `helm`.

**Offered API**

Resource Selector does not export any API.

**Integration with Kubernetes**

Resource Selector is composed by a Kubernetes Deployment and related Service. Relevant files for its deployment can be found on the DECENTER gitlab³.

2.1.6. Resource Seller

A detailed description of the Resource Seller component can be found in D3.4.

**Installation**

Resource Seller can be deployed using `helm`.

**Offered API**

Resource Selector does not export any API.

**Integration with Kubernetes**

Resource Seller is a Kubernetes Deployment. Relevant files for its deployment can be found on the DECENTER gitlab⁴.

2.1.7. IoT Platform

During the project sensiNact and Thingplus have been used as the main IoT Platforms.

The necessary installation steps to deploy sensiNact IoT platform on Kubernetes are reported in D5.1 (Section 2.1.4.1), while the details of the Kubernetes resource file can be found in D3.4 (Section 2.1.4.1).

It is worth mentioning that the docker image of sensiNact has been updated in order to include the semantic layer (described in D4.5) of the Digital Twin.

The details about the integration of Thingplus are exposed below.

---

³ Accessible only to logged in users: https://gitlab.fbk.eu/decenter/w3t1/rs-gui
⁴ Accessible only to logged in users: https://gitlab.fbk.eu/decenter/w3t1/resource_seller
Installation

Thingplus can be installed as from a container repository in any kinds of Kubernetes infrastructures. To verify the installation, you can access the service URL http://192.168.191.149:30655/ on the web browser. The login UI will be shown in they are installed successfully.

![User Graphical Interface Log in.](image)

Offered API

Thingplus provides various APIs to manage services, update attributes, and process timeseries data. Three categories of REST APIs are available in Thingplus as follows:

- Administration REST API - The server-side core APIs.
- Attributes query API - The server-side APIs provided by Telemetry Service.
- Timeseries query API - The server-side APIs provided by Telemetry Service.

Integration with Kubernetes

Thingplus can be deployed on any Kubernetes infrastructure including AWS EKS. The Resource files for the Thingplus pod are provided as follows:

- tp-service.yaml
- tp-deployment.yaml
- tp-pv-and-pvc-nfs-data.yaml
- tp-pv-and-pvc-nfs-db.yaml
2.2. Final DECENTER Application Services, Methods and Tools integration

This section describes the DECENTER’s functionalities and services provided by the DECENTER platform and the DECENTER activities and interfaces to integrate them. As in the previous section, only the new ones and the updates are described.

2.2.1. AI Package

DECENTER provides an AI package which helps making AI microservice from an AI model. The AI package provides functionalities to create an AI model and encapsulate it into an AI method. It provides RESTful APIs to configuration and control of an AI microservice. More details can be found on D4.1, D4.3 and D4.4.

Installation

The package can be found at the following location. The package is written in Python, and can be installed as a Python Package.

- [https://github.com/seungwookti/decenter-ai-package](https://github.com/seungwookti/decenter-ai-package)

To build the Python Package, run this command on the cloned or copied local repository.

```bash
# python3 setup.py bdist_wheel
```

To install the built package, run this command on the same folder.

```
# pip install decenter
```

Offered API & Interfaces for AI microservices

Details of the AI package are described in D4.1 and D4.3. Basically, the AI package provides methods to control AI microservice with uniform methods. These methods include setting location of media input to be analysed to a microservice, setting destination location of analysis result to be delivered, and starting/stopping AI computation. The methods are provided as RESTful APIs.

A user can simply extend the APIs for its own purpose by implementing interfaces to control its own AI microservice. For details, please refer to D4.1 and D4.3.

Integration with DECENTER platform

The purpose of this AI package is to build an AI microservice which is suitable to be deployed on DECENTER platform. DECENTER platform can have heterogeneous resources such as Intel-based VM and/or ARM-based bare metal nodes, and corresponding microservices should be deployed on proper resources. To this end, DECENTER platform has defined a few custom labels to identify AI capabilities of each node. This integration is described in detail in D3.3.

- **Docker image(s)**

  This AI package is to be integrated in a container (AI microservice), and there is no dedicated docker images. This package can be installed on any user-defined docker images.
Kubernetes resource files

ConfigMap resource files are used for configuration of DECENTER AI microservice with DECENTER AI package. DECENTER AI package can configure the basic three components to run an AI service: input, output and the model file locations. The example of ConfigMap is given below, and more details on configuring AI microservice can be found on D4.2.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: uc4-fd-config
data:
  appconfig: |
    
    "input": {
        "url": "http://decenter.keti.re.kr"
    },
    "output": {
        "url": "mqtt://uc4-mqtt:1883/face_image"
    },
    "ai_model": {
        "url": "http://decenter.keti.re.kr/a.zip"
    },
    "autostart": {
        "value": "True"
    }
```

Steps to deploy

- Prepare an AI microservice.
- Prepare an AI model and register it to the AI model registry.
- Prepare ConfigMap resource files. Input, output and model location are written on the ConfigMap.
- Prepare Deployment resource files. Location of container and resource description are written on the Deployment file.
- Apply ConfigMap and Deployment to the Kubernetes master.

2.2.2. Data Management

The Cross-border Data Management scenario that is researched and developed within the DECENTER project aims at allowing participating entities (i.e. users, legal authorities and etc.) to control the data processing and management when it is shared across different administrative domains.

The AI model repository is a tool for managing AI model for DECENTER platform. It provides methods to register and query AI model, and combined with AI package, it can provide efficient way to manage AI models on container runtime. More details can be found on D4.2 and D4.4.

Installation

The AI Model Repository can be installed with Docker container. The source files and dockerfile can be found at the following location:

https://github.com/seungwooketi/ai-model-repository
Basically, this AI Model Repository is based on Flask implementation, and as all the other implementations with Flask, we recommend to use it with other web service for the production level service. The code on the repository is built with NGINX.

To run the AI model repository, simple run model_server.py with Python

```python
# python3 model_server.py
```

After then, you can access the service via web browser with port 5000.

The Data Management microservices can be deployed and run using their Docker container. The source code and the Dockerfile are available in DECENTER's GitLab repository: https://gitlab.fbk.eu/decenter/w4t3/cross-border-data-management

The individual software components of the Cross-border Data Management use the following technologies:

- WebGUI is built following the latest trends: HTML, CSS and JavaScript empowered with React.js.
- ETH Node is used for validating transactions and blocks on the Ethereum blockchain. It can be either running as a service or can be triggered through API enabling faucets.
- LINK Nodes operate between the blockchain and external data, hence allowing Smart Contracts to fetch off-chain data in trustworthy manner.
- Blockchain Service is used to deploy and execute Smart Contracts and facilitate communication with the Smart Oracles. It is based on Node.js implementation and utilizes the web3.js collection of libraries to facilitate interaction with Smart Contracts.
- Trusted Model Manager service collects user input (e.g. AI input data and required AI model) and requests access to the AI model through the Blockchain Service. It is a Java-based component that runs on Apache Tomcat server.

**Offered API & Interfaces for AI microservices**

The AI Model Repository provides RESTful interfaces to register and search AI model of interest. The AI model can be registered by the GUI of the AI Model Repository with its metadata description. The model of interest can be queried and downloaded via HTTP RESTful interfaces with metadata description. Total four kinds of metadata description is needed to describe a model on an AI model repository.

- `model_name`: name of the model
- `model_version`: version of the model
- `model_split`: whether the model is splitted
- `split_number`: index of the splitted model.

An AI model can be registered via web user interfaces of the AI Model Repository. AI Model repository provides RESTful API to access the model of interest with combinations of the metadata above. For example, if you want to find a model with a name of UC4_FaceDetection, version 0.1 and not splitted, the REST url is as follows.

```plaintext
http://localhost:5000/model_download?model_name=UC4_FaceDetection&model_version=0.1&model_split=Split_No&split_number=0
```

**Integration with DECENTER platform**
● **Docker image(s)**

The AI Model Repository is a stand-alone service in DECENTER platform. The dockerfile to build an image can be found on the same location. To run the container of AI model repository, you can simply use docker-compose.

```
# docker-compose -d up -build
```

● **Kubernetes resource files**

No specific resource files are required to deploy the AI Model Repository.

● **Steps to deploy**

No specific steps are required to deploy the AI Model Repository.
3. Final Integration of Use-case AI Applications

All the use case applications for DECENTER have been integrated within the project’s Gitlab repository, under a common path: https://gitlab.fbk.eu/decenter/w5t1. Each use case has its own folder, where the corresponding code, Dockerfiles and Kubernetes resource files have been uploaded. In addition, the Docker images are uploaded in a common Docker Container Registry handled by DECENTER’s Gitlab in: https://gitlab.fbk.eu/decenter/decenter-images/container_registry. Having this common registry allows each use case pilot to deploy the project’s applications seamlessly.

This section described the updates about the integration of the DECENTER AI Application into the use cases.

3.1. Smart City Crossing Safety

This use-case was proposed to show how the technological advances of edge-computing and fast redeployment of containerised AI services proposed by the DECENTER project can be used to improve the safety of pedestrian crossings in future smart city scenarios. For this purpose, a number of components were designed, developed, tested and containerised within WP3 and WP4 activities before these could be then deployed for validation on a first pedestrian crossing by Y2, with the experimentation extended to a second crossing by Y3.

3.1.1. Integration with DECENTER Platform Services

As already reported, UC1 uses several AI Services involving object detection and classification of data coming from external endpoints such as IP Cameras, Microphones and IoT devices. Data are fed to a service for the detection of dangerous situations and a service for actuating one or more alerting devices. The architecture also involves an MQTT broker integrated as a means to add additional sources of data and a user control interface which was further developed and made available through a Web interface for easy consultation of the running service from the Municipality of Trento and for immediate troubleshooting purposes.

Few tests were executed to assess the advantages of edge deployment compared to having some algorithms running in the backend cloud. Being all services deployed on a cluster managed by the DECENTER Fog platform across cloud and edge tiers allowed the assessment of proposed platform value for this time of applications, where time to react of the system must be suitable for a timely intervention on the actuators (buzzer and light “imminent danger” signals for pedestrians and cars).

3.1.2. Integration with DECENTER Application Services, Methods and Tools

The modularity of the DECENTER architecture has enabled us to design the implementation in a flexible manner that allows for a wide applicability in different smart crossing contexts and according to diverse infrastructure characteristics and regulations. In particular, we exploited the separation between backend cloud and edge cloud virtual machines, the containerization of microservices and the orchestration features of DECENTER to show how the execution of
The use case can be constantly improved or adapted to the changing characteristics of the hardware and software available at any point in time.

The use case relies on monitoring devices and each type of data source is connected to a container that processes the continuous stream producing discrete events associated with the recognition of an object or a person at a particular time. To ease integration with other DECENTER Application Services, such events are dispatched to the containers that include the models through an MQTT Broker. The events are also dispatched to the container that enables a Digital Twin GUI representation.

The application on each AI Service was written in Python and developed by extending the DECENTER AI Package. The DECENTER Model Repository was used to store and download the AI Model used by each AI Service. Using the DECENTER Fog Platform allowed the use of different AI Models as well as an easy redeployment of services.

Besides the GUI representation, this use case also makes use of the Digital Twin application service provided by the DECENTER Platform to target real-time representations of some IoT elements of the system. Such implementation choice was made to enable the exploitation of the huge diversity of existing devices that might already be deployed in different crossings in different cities. In this context, the Digital Twin gives the possibility of exploiting additional IoT data, which might be useful in training AI algorithms to improve the quality and relevance of alerts and notifications.

3.1.3. Integration of AI Services

AI services and used algorithms include those for detection of objects from video and processing of audio feeding the system for risk situation assessment within the time constraints imposed by the type of application (i.e. remain below 100ms latency in response time). The AI services that have been integrated within the application and developed by using the DECENTER AI package are reported here for completeness though emphasis has been given to duplicate the installations made in a second crossing within the town of Trento, in a new location with a slightly different crossing geometry and increased traffic intensity.

<table>
<thead>
<tr>
<th>AI Service</th>
<th>Uses ML</th>
<th>Needs retraining</th>
<th>Working on streaming data</th>
<th>Uses DECENTER AI Package</th>
<th>Uses ML framework</th>
<th>Uses GPU</th>
<th>AI optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Detection</td>
<td>Yes: CNN, YOLOv3</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>Yes</td>
<td>Yes: OpenCV DNN (Python)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Audio Recognition</td>
<td>Yes: SVM</td>
<td>Yes: with real data (during pilot installation)</td>
<td>Yes: Audio streaming</td>
<td>Yes</td>
<td>Yes: pyAudioAnalysis (Python)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 2. AI Services Integrated.

<table>
<thead>
<tr>
<th>Risk Situation Detection</th>
<th>No: rule-based inference</th>
<th>N/A</th>
<th>Yes: Event streaming</th>
<th>N/A</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
</table>

The models applied on this use case are not optimized since they fit to the resources on the testbed. The applicable AI optimization methods for the AI models are pruning, Model partitioning will have no effect since the model is already fit to the resource of a node. Binarization or Quantization is not suitable since they do not use GPU.

3.2. Robotics Logistics

The UC2 application has been designed to improve the logistics of a system of robots by making use of the DECENTER platform. In order to do so, the UC2 integrates with several DECENTER platform services (built in WP3) and DECENTER Application Services (built in WP4). In this regard, UC2 has prepared several Docker images and Kubernetes resource files that are needed to launch their application successfully within DECENTER’s platform. Those resource files and Docker images are uploaded in DECENTER’s Gitlab Repository and Docker Container Registry, respectively. In addition, the UC2 has developed an AI Service to support their robots on differentiating between human, other robots or anything else, by making use of DECENTER’s AI Package. This distinction opens up a range of possibilities for action when designing robot decision-making.

The original design of the UC2 AI Application can be found in D4.3 and D5.1, which details how the Use Case works by means of several diagrams.

3.2.1. Integration with DECENTER Platform Services

In order to launch the AI Application successfully, the UC2 needed to integrate the following DECENTER’s platform services:

Integration of L-ADS

The L-ADS asset provides security on the microservices in the Robotnik’s use case. The L-ADS is integrated with the rest of microservices. It is able to collect the network traffic from the microservices with the help of the tool called Softflowd.

Once the L-ADS is deployed, it receives the flows from Softflowd that it is sniffing the traffic through the rest of services running in the Robotnik’s use case. The L-ADS receives that traffic for two hours. After that, the asset trains the Autoencoder algorithm and it is ready to classify with connections are benign or anomalous. However, we are not expecting to receive any connection anomalous in this use case. That is the reason to use a script that can modify the values of the connection with the aim to offer how the L-ADS predict those modified connections.

The following figure shows the workflow in the Robotnik’ use case.
Integration of SLA Manager

The Robotnik’s use case makes use of the SLA Manager and SLA GUI components to assess and check the availability of third-party services needed by the main application, following the service level agreement defined by the client (Robotnik) and the provider. In this case the service whose availability needs to be assessed is a service provided by Robotnik itself, but the idea is to extend this functionality to third party services provided by others.

After this service is launched, one of the Logistics Robotics microservices calls the SLA Manager REST API to create an SLA responsible for the assessment of the availability of this service. Then, the SLA Manager is continuously checking and assessing the availability of this service. If it detects any violation, this information is stored in the internal database of the SLA Manager. If needed this information can be sent to external applications responsible for processing these violations. These external applications only need to subscribe to the service responsible for sending these violations or notifications to get them. In this case, ATOS just provides a Graphical User Interface to get all this information, and to see if there are any violations of the SLA.

Next figure depicts the sequence diagram of the integration between the SLA Manager and the Robotnik’s use case.
3.2.2. Integration with DECENTER Application Services, Methods and Tools

As explained previously, the UC2 offers one AI service built within DECENTER’s scope. In order to build this AI service, DECENTER’s AI Package was used.

The Object Detector AI service uses the AI Package in several ways. Mainly, the AI Package allows to serve the trained model as a REST service. It offers an API that handles the usage of the model seamlessly. In addition, the version of the model that is served is also configured by means of the AI Package functionalities. The whole configuration of the AI Package is
passed to the Docker Image of the AI Service as an environment variable named “appconfig”. More information on how the AI Package is used can be found in D4.3 and D4.4.

3.2.3. Integration of AI Services

The development of this AI service was done within DECENTER's Gitlab repository and has followed the CI/CD guidelines.

As shown in Table 4: UC2 AI Service, this AI service uses the pre-trained YOLOv3\textsuperscript{5} algorithm with the Tensorflow AI framework to train a new version of the model with data generated within UC2, to detect humans and robots from the point of view of the built-in camera of those robots.

There were built two versions of this AI Service, one from the simulation environment and one from the real environment. Both environments' final datasets included more than 400 images of humans and robots each. Those datasets are also uploaded within the DECENTER’s Gitlab premises.

<table>
<thead>
<tr>
<th>AI Service</th>
<th>AI Algorithm</th>
<th>Retraining</th>
<th>AI Framework</th>
<th>Uses GPU</th>
<th>DECENTER AI Package</th>
<th>AI optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Detector</td>
<td>YOLOv3</td>
<td>Yes: UC2 prepared dataset</td>
<td>Tensorflow</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4. UC2 AI Service.

The AI optimization has not been applied to this use case since the model is already fit to the computational resources. The suitable AI optimisation methods for this use case can be model pruning, since it can be run on generic hardware.

The integration of this AI service starts from the Gitlab repository, where the code, that uses the DECENTER's AI Package to load and serve the trained model, can be found. Once the application is built as a Python software, it is integrated into a Docker image and uploaded into DECENTER’s Gitlab Container Registry.

In order to deploy the AI Service in DECENTER's platform, the required Kubernetes resource files were built. An example of how those files look is shown below but they can be found in DECENTER’s Gitlab Repository at https://gitlab.fbk.eu/decenter/w5t1/uc2-logistics-robotics-optimization/object-detector-yolov3/-/tree/master/deployment/k8s. As it can be seen in the following examples, these files use DECENTER's Image Registry to load the specific Object Detector image.

**Deployment resource file**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: object-detection-deploy
```

\textsuperscript{5} https://github.com/wizyoung/YOLOv3_TensorFlow
labels:
  app: decenter-usecase2
decenter-service-type: ai-service
decenter-service: object-detection
spec:
  replicas: 1
  selector:
    matchLabels:
      app: decenter-usecase2
decenter-service-type: ai-service
decenter-service: object-detection
strategy:
  type: Recreate
template:
  metadata:
    labels:
      app: decenter-usecase2
decenter-service-type: ai-service
decenter-service: object-detection
spec:
  containers:
  - name: object-detection-image
    image: gitlab-registry.fbk.eu/decenter/decenter-images/object-detector-yolov3:1.3.0
    ports:
      - containerPort: 5000
    env:
      - name: MY_APP_CONFIG
        valueFrom:
          configMapKeyRef:
            name: object-detection-config
            key: appconfig
    # imagePullSecrets: regcrec
    stdin: true
tty: true
  restartPolicy: Always

Service resource file

apiVersion: v1
type: Service
kind: Service
metadata:
  name: object-detection-service
labels:
  app: decenter-usecase2
decenter-service-type: ai-service
decenter-service: object-detection
spec:
  ports:
  - name: "compute-port"
    port: 5000
    targetPort: 5000
    nodePort: 30000
  selector:
    decenter-service: object-detection
type: NodePort

ConfigMap resource file

apiVersion: v1
type: ConfigMap
kind: ConfigMap
D5.2 Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations

```yaml
metadata:
  name: object-detection-config
data:
  appconfig: |
    "input": {
      "url": ""
    },
    "output": {
      "url": {
        "detected_cor": "mqtt://test.mosquitto.org:8080/DECENTER/UC2/object-detector"
      }
    },
    "ai_model": {
      "url": "http://127.0.0.1:5000",
      "model_name": "decenter_yolov3",
      "model_version": "0.4"
    },
    "autostart": {
      "value": "False"
    }

These files are deployed easily with:

```
kubectl apply -f configmap.yaml -f service.yaml -f deployment.yaml
```

3.3. Smart and Safe Construction

DECENTERT Use Case 3: Smart and Safe Construction is designed and implemented to bring AI intelligence to the constructions sites to increase the level of on-field safety. The use case utilises AI models for object detection and member verification. Essentially, the object detection model allows for timely detection of different objects on the construction site, whereas the member verification model facilitates the detection and identification of construction personnel. Because the use case uses member verification AI models that rely on biometric and private data, the use case integrates with the components developed in the scope of cross-border data management. In the following sections we will describe the details of the final integration of the use case with DECENTER.

3.3.1. Integration with DECENTER Platform Services

The Smart and Safe construction site use case is designed and developed to utilise and communicate with the DECENTER components. The integration plan with the DECENTER Platform Services and UC-3 workflow of events was described in D5.1. In particular, to extensively use the (re)deployment mechanisms and QoS runtime management, UC-3 is integrated with the following platform services:

1. Integrated with the Fog Platform Orchestrator: The MDP (re)deployment mechanism that has been aligned with WP2 Resource Models and it is used for QoS-aware orchestration of AI models.
2. Integrated with Prometheus monitoring system of the DECENTER Platform: The QoS-aware orchestration of AI models is based on monitoring data received from the monitoring system.

Integrated with IoT Platform Dashboard/Manager/Edge in order to gain access to IoT sensor data.

Moreover, the use case exploits the benefits of the SLA Manager (i.e. management of SLAs, detection of violations and service payment), Application composer (i.e. select AI model, sensor data and etc.) and Service Discovery (i.e. detect available deployment options, detect AI models, detect available sensors and etc.). In Y3, the use case was adapted to utilise Smart Contracts for SLA management.

3.3.2. Integration with DECENTER Application Services, Methods and Tools

UC-3 is integrated with two services: Data Management [D4.4] and the AI Package serving system [D4.3].

To apply policies and regulations on the AI-model utilisation, the Smart and Safe construction site integrates with the cross-border data management components. Hence, the AI-models could be deployed and run only upon successful execution of the blockchain transaction (i.e. Smart Contract function). In Y3, the Smart Contracts were optimised for better performance. Furthermore, the use case implemented the Smart Oracles from the Data Management service, to facilitate trustworthy data to the Smart Contracts.

3.3.3. Integration of AI Services

UC3 integrates three AI services that were developed within WP4 [D4.4]. In particular the use case integrates the following AI services: Object detection (to detect vehicles, tools, equipment at the construction site) and Face Detection/Recognition (to perform member verification and detect intruders at the construction site).

The following services were developed complementary by UL: Mask Detection (to detect if a person wears a face mask for COVID-19 protection), Vehicle license plate detection/recogniton (to verify if the vehicle is allowed on the construction site premises), Weather classification (identify the current weather status and notify for forthcoming storms). The AI services have been integrated within the Smart and Safe construction use-case and are described in Table 5.

---

The AI optimisation methods are not applied to the models in this use case since resources on this use case are already sufficient to run those models. For the GPU-enabled models, binarization can be applied for the optimisation. For the other models, model pruning can optimize the resource utilization.

### Ambient Intelligence for Office Environments

#### Integration with DECENTER Platform Services

Ambient Intelligence for Office Environments is composed with 4 AI services: face detector, feature extractor and two member-verifiers. DECENTER Platform has been used for deploying those microservices on a cluster, monitoring the status of them and managing AI models.

<table>
<thead>
<tr>
<th>AI Service</th>
<th>Uses ML</th>
<th>Needs retraining</th>
<th>Working on streaming data</th>
<th>Uses DECENTER AI Package</th>
<th>Uses ML framework</th>
<th>Uses GPU</th>
<th>Uses GPU</th>
<th>AI optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Detection</td>
<td>Yes: YOLOv3</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>Yes</td>
<td>Yes: OpenCV DNN</td>
<td>Yes, if present</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Face Detection</td>
<td>Yes: CNN, YuFaceDetectnet</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Face Recognition</td>
<td>Yes: DNN-based</td>
<td>Yes</td>
<td>Yes: Video streaming</td>
<td>No</td>
<td>Yes: Scikit-learn</td>
<td>Yes, if present</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Mask Detection</td>
<td>Yes: YOLOv5</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>Yes</td>
<td>Yes: TensorFlow, OpenCV</td>
<td>Yes, if present</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>License plate detection/recognition</td>
<td>Yes: YOLOv3</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>Yes</td>
<td>Yes: TensorFlow, OpenCV</td>
<td>Yes, if present</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Weather classification</td>
<td>Yes: CNN</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>No</td>
<td>Yes: TensorFlow, OpenCV</td>
<td>Yes, if present</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Table 5. AI Services Integrated.
The Orchestrator of uc4 is integrated using the service named application composer of FogPlatform. It offers efficient GUI interface to make it easy deployment of AI services on an edge cluster [Figure 4]. The Monitoring System of Ambient Intelligence for Office Environments shows the location of the devices that have each microservice and the status of containers in real time [Figure 5]. Orchestrator and Monitoring System are connected via
FogPlatform. FogPlatform containers are integrated with KETI testbed by deploying them on that testbed and opening the specific port number to the monitoring system.

Figure 6. Dashboard of UC4: the status of deployed containers.

Figure 7 Dashboard of UC4. monitoring of AI services output and processing time.

The Dashboard is a web service that shows the status of deployed containers [Figure 6], the details of outputs from all AI services and service processing time [Figure 7]. It uses a specific UC4 domain name for connecting with other AI services on a testbed to request changing an AI model in a container when it be needed. The domain name is transmitted by one parameter.
named ‘uc4_domain’ in configmap yaml file. The configmap of the dashboard can be shown as below. To access to other AI services using ‘uc4_domain’, each AI service need to define their hostname in default-subdomain. For that reason, the deployment file of each AI service has to include the hostname definition as shown the Deployment resource file below.

**ConfigMap resource file**

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: uc4-mnt-config
data:
  appconfig: |
    "mqtt_config": {
      "mqtt_broker_url": "mqtt-broker.default-subdomain.default.svc.cluster.local",
      "mqtt_broker_port": 1883
    },
    "fd_config": {
    },
    "ip_cam_config": {
      "ip_cam_url": "http://ketihhc.iptime.org:5004/mjpg/1/video.mjpg"
    },
    "domain_config": {
      "uc4_domain": "default-subdomain.uc4.svc.cluster.local:5000"
    }
```

**Deployment resource file**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fd-deployment
  namespace: uc4
labels:
  name: face-detector
spec:
  replicas: 1
  selector:
    matchLabels:
      name: face-detector
  template:
    metadata:
      labels:
        name: face-detector
      app: uc4
    spec:
      securityContext:
        fsGroup: 65534
      hostname: face-detector
      subdomain: default-subdomain
      containers:
        - name: face-detector
          image: keticmr.iptime.org:22500/uc4-fd:kpi1.1
          imagePullPolicy: Always
          volumeMounts:
            - mountPath: "/log"
```

3.4.2. Integration with DECENTER Application Services, Methods and Tools

Digital Twin

For integration with Digital Twin, we defined new data type transferring to Digital Twin. MQTT message broker get TYPE_MR and TYPE_FI_DT data from UC4-FE(feature extractor) and UC4-MV (member verifier).

Data examples are as follows:

```python
(UC4-FE: TYPE_FI_DT)
{'timeStamp': '2020-07-03 08:10:10', 'cord': [0.4431998133659363, 0.4868328273296356, 0.699257493019104, 0.742890477180481]}
{'timeStamp': '2020-07-03 08:10:11', 'cord': [0.45010021328926086, 0.2947275638580322, 0.701793909072876, 0.546421229839325]}
(UC4-MV, TYPE_MR)
```
Table 6. Face_image_dt format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Datetime (%Y-%m-%d %H:%M:%S)</td>
<td>2020-03-20 03-02-25</td>
</tr>
<tr>
<td>Face_position</td>
<td>float16 x4 (Lefttop_x, Lefttop_y, Rightbottom_x, Rightbottom_y)</td>
<td>100, 80, 160, 160</td>
</tr>
</tbody>
</table>

Table 7. Member_result format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Datetime (%Y-%m-%d %H:%M:%S)</td>
<td>2020-03-20 03-02-25</td>
</tr>
<tr>
<td>Result</td>
<td>String (Yes</td>
<td>No)</td>
</tr>
<tr>
<td>GroupName</td>
<td>String</td>
<td>GroupA</td>
</tr>
<tr>
<td>Confidence</td>
<td>Float</td>
<td>0.92</td>
</tr>
</tbody>
</table>

**AI Package and Data optimization method**

UC4 is also integrated with AI package and data optimization method. All Optimized AI models are stored in AI model repository. Further details are described in D4.
### 3.4.3. Integration of AI Services

This application checks the face of users visiting a certain space and verifies whether the person is authorised to consume certain content in that space or not. It integrated two verifiers that can verify two group members: A group verifier, and B group verifier. We assume that only these two groups are targeting to see specific content.

The deployment makes proper use of edge and cloud resources to ensure user face image and model privacy. Thus, the processes at the edge can verify whether the visitor of a certain space can consume certain content or not in that space, without sharing personal information with the cloud.

<table>
<thead>
<tr>
<th>#</th>
<th>AI Application</th>
<th>AI application service</th>
<th>Activity kind</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Content Access Authorization</td>
<td>Face Detection (FD)</td>
<td>Analysis</td>
<td>When US MS requests, FD starts detecting a face in the image from camera. When it detects the face, it crops face area and transfers it to MV MS.</td>
</tr>
<tr>
<td>2</td>
<td>Content Access Authorization</td>
<td>Member Verification (MV)</td>
<td>Analysis</td>
<td>This module verifies that person who is in the image from FD MS is X group member or not. This module gets the X group MV AI model from AI repository.</td>
</tr>
<tr>
<td>3</td>
<td>Content Access Authorization</td>
<td>Service Control</td>
<td>Decision</td>
<td>It gets the results from Member Verification microservices and checks whether this person is the target member. In this case, it authorizes it to consume the content.</td>
</tr>
<tr>
<td>4</td>
<td>Content Access Authorization</td>
<td>Content Serving</td>
<td>Action</td>
<td>If the detected person is analyzed as a target member, the service provides the appropriate content to the service front-end.</td>
</tr>
</tbody>
</table>

Table 8. Functionalities of AI Applications.
UC-4 integrates three AI services that were developed within WP4. In particular the use case integrates the following AI services: Facedetector, facialfeature extractor and 2 group member verifiers.

Input and output of AI services are as follows.

<table>
<thead>
<tr>
<th>Module Name</th>
<th>input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face Detection</td>
<td>640X640 (TBD) image data Frame rate: under 10 f/s</td>
<td>Cropped face image having only one human face, Time, face size</td>
</tr>
<tr>
<td>Face Feature Extractor</td>
<td>Cropped face image (180x120*3 Ch)</td>
<td>Extracted features (512, 1)</td>
</tr>
<tr>
<td>Group MV</td>
<td>Extracted features(512,1)</td>
<td>Yes/No</td>
</tr>
</tbody>
</table>

Table 9. Inputs and outputs of AI services.

We defined 4 data types for connection with other microservices: TYPE_FI, TYPE_FF, TYPE_MR, TYPE_FI_DT.

<table>
<thead>
<tr>
<th>Type name</th>
<th>Data Value Type</th>
<th>Format/Size</th>
<th>Common Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_FI</td>
<td>Image</td>
<td>UInt8/(160, 160, 3)</td>
<td></td>
</tr>
<tr>
<td>TYPE_FF</td>
<td>Feature Set</td>
<td>Float32/(512, 1)</td>
<td>Time_Sstamp Datetime</td>
</tr>
<tr>
<td>TYPE_MR</td>
<td>Result</td>
<td>String/(Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>GroupName</td>
<td>String/(under 10 bytes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Confidence</td>
<td>Float(1)</td>
<td></td>
</tr>
<tr>
<td>TYPE_FI_DT</td>
<td>Face_Position_uint8</td>
<td>UInt8(4, 1)</td>
<td></td>
</tr>
</tbody>
</table>

Table 10. Types of connection with other microservices.
And each data type can be input or output of micro service. The following table shows the input, input type, output and output type of microservices.

<table>
<thead>
<tr>
<th>Identifier (MS)</th>
<th>Input</th>
<th>Input Type</th>
<th>Output</th>
<th>Output type</th>
<th>Base Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC4-FD</td>
<td>URL of Camera (HTTP Get)</td>
<td>Video Streaming</td>
<td>Mqtt://UC4-mqtt/face_image</td>
<td>TYPE_FI</td>
<td>DECENTER-AI</td>
</tr>
<tr>
<td>UC4-FE</td>
<td>Mqtt://UC4-mqtt/face_image</td>
<td>TYPE_FI</td>
<td>Mqtt://UC4-mqtt/face_feature</td>
<td>TYPE_FF</td>
<td>DECENTER-AI</td>
</tr>
<tr>
<td>UC4-MV (A &amp; B)</td>
<td>Mqtt://UC4-mqtt/face_feature</td>
<td>TYPE_FF</td>
<td>HTTP Post</td>
<td>TYPE_MR</td>
<td>DECENTER-AI</td>
</tr>
<tr>
<td>UC4-CS</td>
<td>TYPE_MR</td>
<td>TYPE_MR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UC4-MQTT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Mosquito</td>
</tr>
</tbody>
</table>

Table 11. Specifications of microservices.

The following table shows the AI services integrated in UC4 and its specifications:

<table>
<thead>
<tr>
<th>AI Service</th>
<th>Uses ML</th>
<th>Needs retraining</th>
<th>Working on streaming data</th>
<th>Uses DECENTER AI Package</th>
<th>Uses ML framework</th>
<th>Uses GPU</th>
<th>AI Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC4-FD</td>
<td>Yes: CNN DSFD</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>Yes</td>
<td>Yes: OpenCV, Tensorflow (Python)</td>
<td>Yes</td>
<td>Yes: Pruning, Quantization</td>
</tr>
<tr>
<td>UC4-FE</td>
<td>Yes: CNN</td>
<td>No</td>
<td>Yes: Video streaming</td>
<td>Yes</td>
<td>Yes: Tensorflow (Python)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>UC4-MV</td>
<td>Yes: CNN, classifier</td>
<td>Yes: with real data of members</td>
<td>No</td>
<td>N/A</td>
<td>Yes: Tensorflow (Python)</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 12. AI Services Integrated.
4. Final Demonstrators and pilots

This section covers the final updates related with the definition and development of the four demonstrators and use-case pilots involved within the project. In order to make KPIs clearer and to be able to show quantitative and qualitative aspects, the KPIs of UCs have been revised and updated. The final and detailed version of them is attached to the annex of this document. Also, the description about the demonstration of each KPI is included as well as the results and their analysis per use-case.

Since, a part of use-case KPIs, there are KPIs related with the DECENTER application service of Digital Twin, this section starts with the specification of DT KPIs individually and then, their demonstration and results have been described inside use-case 1 and 4 subsections, where it was integrated and tested.

4.1. Digital Twin KPIs

For the purpose of demonstrating, with measurable values, how effectively the DT is achieving its objectives, a set of KPIs have been designed and implemented. These KPIs assess the most important functionalities of the DT, as well as the most important aspects of its semantic layer. The semantic layer, built on the top of the DT, offers a consistent way of interpreting data from any UC and it is easily understandable by humans and machines. The semantic layer consists of a general-purpose IoT ontology (i.e., sensiNactOntology), a domain specific (i.e., SmartCrossing) ontology and a rule-based reasoner for extracting new knowledge. A detailed presentation of this work can be found in D4.5.

The table below summarizes the KPIs for the DT and the semantic layer. For each KPI, a description is provided, an appropriate measure for quantifying it as well as the methodology being used for obtaining the value of this measure.

<table>
<thead>
<tr>
<th>No</th>
<th>KPI Name</th>
<th>KPI Description</th>
<th>Measure</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DT2</td>
<td>Assets Integration</td>
<td>Number of integrated assets (e.g., physical/digital entities, AI models)</td>
<td>For each UC, calculate the number of AI entities, IoT devices and other digital entities being represented by the DT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Demonstrates the set of integrated assets (physical/digital entities) and the addressed interoperability issues</td>
<td>Number of communication protocols being used</td>
<td>For each UC, provide the number of protocols being used in order to enable the communication between the UC and the DT.</td>
</tr>
<tr>
<td></td>
<td>DT2</td>
<td>Data Integration</td>
<td>Average number of messages per hour per UC.</td>
<td>Measure the average number of messages per hour, being received by each UC, during a day period.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Measures the amount of data retrieved and represented by the</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In the next subsections (4.2.2.a, 4.5.2.a), the results of the KPIs evaluation for UC1 and UC4 have been presented.

### 4.2. Smart City Crossing Safety

On 22 June 2020, all the devices had been installed on the pedestrian crossing in the first pilot area in the neighbourhood of Vela, Trento. The deployment of a second pilot in a nearby neighbourhood started in March 2021 and is currently ongoing at the time of writing this deliverable. The second pilot is being installed in an area that has different characteristics (e.g. heavier traffic) and will be finalized before the end of the project.

#### 4.2.1. MVD

The Minimum Viable Demonstrator for this use case was meant to show how DECENTER technology advances could be exploited in order to improve the safety of pedestrian crossings.
in future Smart Cities. The demonstrator highlighted advantages of having AI solutions and algorithms running close to where the monitoring and sensing normally take place. The main requirements underpinning the implementation were related to having to ensure the availability of internet connectivity to the roadside cabinet where all the edge devices are installed. With a proper network reaching the roadside cabinet for backend cloud connectivity, an MVD can be configured with an edge computing device (rugged PC with industrial grade performance for harsh environments - we used a barebone PC (i7 6 core/12 thread, 16GB ram, 256 SSD) and a backend cloud (two Virtual Machines with 2 CPUs and 2GB of RAM)). We added additional hardware elements for actuating lights and buzzers for alerting both pedestrians wanting to cross the road and vehicles approaching the crossing. The MVD can also include microphones for listening to nearby sounds and IoT devices for monitoring weather and environmental conditions. These additional devices are envisaged to provide enough redundancy for robust decision making when it comes to identifying dangerous situations through what is interpreted exclusively through the cameras’ video interpretation. The figure below highlights all these components.

**Crossing Safety – Application Components**

The running application components are containerised and they expose interpreted events on the MQTT channel; these events are then consumed by the logic (Risk Situation Detector and Alarm Service) implemented to interpret dangerous situations and actuate alerts accordingly. This modular design allows for replacements of containers, manual and automated repositioning of these between edge and backend cloud according to performance requirements / limitations.

In the following table we illustrate the functional requirements addressed with the use case; it must be noted that, because of the execution workplan, functional requirements were mostly addressed by Y2 whereas the focus for Y3 was on non-functional requirements improving the prototype to perform in a reliable and robust manner.
### Functional Requirement

**FR_UC1_001**
Identify moving vehicles approaching a pedestrian crossing

AI algorithms process raw data inputs from a series of different devices (cameras, microphones, IoT devices) and are trained to identify different vehicle types and speed of approach.

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

**FR_UC1_002**
Identify moving people approaching a pedestrian crossing

AI algorithms process raw data inputs from a series of different devices (cameras, microphones, IoT devices) and are trained to identify people.

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

**FR_UC1_003**
process inferred events with the objective of identifying dangerous situations

Identified events are further processed and fed to a rule-based engine.

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

**FR_UC1_004**
alert pedestrians and drivers of imminent danger

Realised through lights and buzzing actuators.

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

**FR_UC1_005**
identify weather and environmental conditions

The use case experimentation includes also a number of IoT sensing devices (temperature, light, humidity, air and road surface).

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

**FR_UC1_006**
associate different traffic levels with different crossing conditions

The system will behave differently in different traffic conditions, with more alerts when situation is such that an accident is more likely.

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

### Non-functional Requirement

**NFR_UC1_001**
assess system performance degradation

monitoring latency between identification of dangerous events and alerts

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

**NFR_UC1_002**
allow easy moving of containers from the Cloud to the Edge when latency degrades

vertical offloading

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y2</td>
</tr>
</tbody>
</table>

**NFR_UC1_003**
preserve privacy of people involved in the pedestrian crossing

video streaming only available live on the edge computing box

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O any privacy-sensitive content is discarded once events are inferred</td>
</tr>
</tbody>
</table>

**NFR_UC1_004**
be accurate and reactive

Tuning of monitoring devices was performed but

<table>
<thead>
<tr>
<th>Comments on how to address the requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>O Covered at Y3</td>
</tr>
</tbody>
</table>

---

Table 14. List of functional requirements for the UC1.

The following table lists the non-functional requirement characterising the system that realise the smart crossing implementation.
In the following table are described the elements for realising a minimum viable demonstrator:

<table>
<thead>
<tr>
<th>Demo storyboard</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify pedestrian crossing and assess its geometry to decide on positioning of devices</td>
<td></td>
</tr>
<tr>
<td>Installation and deployment on site of devices and services following in-lab installations of bootstrapping code and testing</td>
<td></td>
</tr>
<tr>
<td>tuning of devices for best monitoring</td>
<td></td>
</tr>
<tr>
<td>activation of wanted operational rules according to the operator requirements</td>
<td></td>
</tr>
<tr>
<td>assessment of system performance and positioning of container between edge and cloud is adequate for required prototype execution</td>
<td></td>
</tr>
<tr>
<td>ensure all devices are working well judging correspondence between what monitored through the dashboard and the visual assessment of situations at crossing</td>
<td></td>
</tr>
</tbody>
</table>
4.2.2. KPIs

As it was explained in the introduction of this section 4, for use-case 1 and 4 the KPIs’ section is split in two subsections, one related to Digital Twin and Semantic Layer KPIs and another related to the KPIs specific for the use-case.

a. DT and SL KPIs

All KPIs described in Section 4.1 were implemented and demonstrated in the context of UC1. The table below presents the results of the DT KPIs.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT1: Assets Integration</td>
<td>1 AI (Object Recognition), 3 IoT devices (2 cameras, weather station), 14 Digital Entities (e.g., Crossing, Vehicle, Pedestrian)</td>
</tr>
<tr>
<td></td>
<td>1 communication protocol: MQTT</td>
</tr>
<tr>
<td>DT2: Data Integration</td>
<td>Average number of messages per hour in a day: 543</td>
</tr>
<tr>
<td></td>
<td>Maximum number of messages in an hour of a day: 2389</td>
</tr>
</tbody>
</table>

Table 16. Results of the DT KPIs.

Below we provide the results of the semantic layer KPIs as well as a descriptive analysis of the methodology followed in order to obtain them.

KPI SL1: Completeness

Description of demonstration: For the purpose of quantifying the completeness of the semantic layer, the functional requirements of UC1 (described in D2.2) were contrasted with the existing entities in the ontology. The table below matches the functional requirements with their relevant ontological entities. Functional requirements that are not relevant with data representation (e.g., data backup, redeployment of AI model) are omitted.

<table>
<thead>
<tr>
<th>FR ID</th>
<th>FR description</th>
<th>Semantic representation</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_UC1_001</td>
<td>Detect if a pedestrian is approaching the crossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smart">www.semanticweb.org/ontologies/2021/smart</a> CrossingOntology#Pedestrian, <a href="http://www.semanticweb.org/ontologies/2021/smart">www.semanticweb.org/ontologies/2021/smart</a> CrossingOntology#isAt, <a href="http://www.semanticweb.org/ontologies/2021/smart">www.semanticweb.org/ontologies/2021/smart</a> CrossingOntology#Crossing</td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_002</td>
<td>Detect if a cyclist is approaching the crossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smart">www.semanticweb.org/ontologies/2021/smart</a> CrossingOntology#Cyclist, <a href="http://www.semanticweb.org/ontologies/2021/smart">www.semanticweb.org/ontologies/2021/smart</a> CrossingOntology#isAt, <a href="http://www.semanticweb.org/ontologies/2021/smart">www.semanticweb.org/ontologies/2021/smart</a> CrossingOntology#Crossing</td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_003</td>
<td>Detect if a vehicle is approaching the crossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Cyclist">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Cyclist</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt">www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing</a></td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_004</td>
<td>Detect if a disabled person is approaching the crossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Pedestrian">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Pedestrian</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt">www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing</a></td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_005</td>
<td>Detect if a person with a pet is approaching the crossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#PetOwner">www.semanticweb.org/ontologies/2021/smartCrossingOntology#PetOwner</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt">www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing</a></td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_006</td>
<td>Detect if a person with a stroller is approaching the crossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Stroller">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Stroller</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt">www.semanticweb.org/ontologies/2021/smartCrossingOntology#isAt</a>, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Crossing</a></td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_009</td>
<td>Trigger actuators in case of dangerous situation (i.e. when a RED alert is ON)</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#RedAlertON">www.semanticweb.org/ontologies/2021/smartCrossingOntology#RedAlertON</a>, instanceof, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Event">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Event</a></td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_010</td>
<td>Trigger a sound alarm in case of dangerous situation</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#SoundAlarmON">www.semanticweb.org/ontologies/2021/smartCrossingOntology#SoundAlarmON</a>, instanceof, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Event">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Event</a></td>
<td>SmartCrossing</td>
</tr>
<tr>
<td>FR_UC1_011</td>
<td>Flashing lights in case of dangerous situation (i.e. when a RED alert is ON)</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#FlashingLightsON">www.semanticweb.org/ontologies/2021/smartCrossingOntology#FlashingLightsON</a>, instanceof, <a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#Event">www.semanticweb.org/ontologies/2021/smartCrossingOntology#Event</a></td>
<td>SmartCrossing</td>
</tr>
</tbody>
</table>

Table 17. Functional Requirements with ontology relationship.
Results

The percentage of functional requirements, which are being covered by the ontology, represents the completeness of the semantic layer. The overall completeness of the semantic layer is shown below. Moreover, the table shows how much of the functional requirements are being covered by each one of the two ontologies, sensiNactOntology and SmartCrossing.

<table>
<thead>
<tr>
<th>UC1</th>
<th>sensiNactOntology</th>
<th>SmartCrossing</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>10%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Table 18. Completeness of DT KPI.

KPI SL2: Conciseness

**Description of demonstration:** For the purpose of quantifying the conciseness of the semantic layer, the entities of each ontology were listed in order to identify how many of these entities are related with UC1 domain.

<table>
<thead>
<tr>
<th>sNaOntology</th>
<th><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#">www.semanticweb.org/ontologies/2021/smartCrossingOntology#</a></th>
<th>Include in UC1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td>x</td>
<td>Pedestrian</td>
<td>x</td>
</tr>
<tr>
<td>Service</td>
<td>x</td>
<td>Cyclist</td>
<td>x</td>
</tr>
<tr>
<td>Resource</td>
<td>x</td>
<td>Stroller</td>
<td>x</td>
</tr>
<tr>
<td>Person</td>
<td>x</td>
<td>Petowner</td>
<td>x</td>
</tr>
<tr>
<td>Group</td>
<td></td>
<td>Camera</td>
<td>x</td>
</tr>
<tr>
<td>Location</td>
<td>x</td>
<td>Crossing</td>
<td>x</td>
</tr>
<tr>
<td>RelativeLocation</td>
<td></td>
<td>Vehicle</td>
<td>x</td>
</tr>
<tr>
<td>AbsoluteLocation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 19. Relation between Ontologies and UC1 domain.

Results

The percentage of ontology’s entities also included in the UC1 represents the conciseness of the semantic layer. The table below shows the overall conciseness of the semantic layer, as well as the percentage of relevant entities to UC1 for each ontology.

<table>
<thead>
<tr>
<th>UC1</th>
<th>sensiNactOntology</th>
<th>SmartCrossing</th>
</tr>
</thead>
<tbody>
<tr>
<td>88%</td>
<td>80%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 20. Conciseness of SL KPI.
KPI SL3: Adaptability

Description of demonstration: For the purpose of quantifying the adaptability of the semantic layer, the ontological entities were listed in order to discover how many of them are being reused by both UCs (UC1 and UC4).

<table>
<thead>
<tr>
<th>Entity</th>
<th>Included in UC1 and UC4</th>
<th>Provider</th>
<th>Service</th>
<th>Resource</th>
<th>Person</th>
<th>Group</th>
<th>Location</th>
<th>RelativeLocation</th>
<th>AbsoluteLocation</th>
<th>Event</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sNaOntology</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#">www.semanticweb.org/ontologies/2021/smartCrossingOntology#</a></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sNaOntology</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#">www.semanticweb.org/ontologies/2021/sensiNactOntology#</a></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmartCrossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#">www.semanticweb.org/ontologies/2021/sensiNactOntology#</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmartCrossing</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/smartCrossingOntology#">www.semanticweb.org/ontologies/2021/smartCrossingOntology#</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 21. Ontologies related with UC1 and UC4.

Results

The percentage of ontology’s entities included in both UCs represents the adaptability of the semantic layer to different domains. The table below shows the overall adaptability of the semantic layer, as well as the adaptability of each ontology. It is worth mentioning that the zero adaptability of SmartCrossing is expected, since it is a domain-specific ontology dedicated to UC1 and it is not designed to cover several domains. On the contrary, the general-purpose sensiNactOntology reaches a 70% ability to adapt on different domains (although it may not be able to cover all their different concepts, see KPI SL1).

<table>
<thead>
<tr>
<th>DT2: Adaptability</th>
<th>Overall</th>
<th>sensiNactOntology</th>
<th>SmartCrossing</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1</td>
<td>41%</td>
<td>70%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 22. Adaptability of KPI to UC1.

KPI SL4: Clarity

Description of demonstration: In order to achieve clarity, we documented both ontologies, their entities as well as the relations among their entities. The documentation is presented in D4.5.
KPI SL5: Inferred Knowledge

**Description of demonstration:** In order to quantify the knowledge produced by the semantic layer we count the number of axioms generated by each ontology, as well as the number of classes, objects, data properties and individuals being instantiated for each ontology. The table below summarizes and describes these measurements.

<table>
<thead>
<tr>
<th>Ontology metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiom</td>
<td>The combined logical and non-logical axiom count</td>
</tr>
<tr>
<td>Class count</td>
<td>The number of entities</td>
</tr>
<tr>
<td>Object property count</td>
<td>The number of object properties, describing relations among entities (e.g., hasTemperature)</td>
</tr>
<tr>
<td>Data property count</td>
<td>The number of data properties, describing attributes of an entity (e.g., temperature)</td>
</tr>
<tr>
<td>Individual count</td>
<td>The number of individuals (e.g., SpeedViolationEvent)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ontology metric</th>
<th>sensiNactOntology</th>
<th>SmartCrossing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiom</td>
<td>158</td>
<td>56</td>
</tr>
<tr>
<td>Class count</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Object property count</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>Data property count</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>Individual count</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

*Table 23. Ontology metrics description.*

**Results**

<table>
<thead>
<tr>
<th>Ontology metric</th>
<th>sensiNactOntology</th>
<th>SmartCrossing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiom</td>
<td>158</td>
<td>56</td>
</tr>
<tr>
<td>Class count</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Object property count</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>Data property count</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>Individual count</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

*Table 24. Ontology metric results.*

b. **UC1 KPIs**

**KPI 1.** Detecting key actors involved in the pedestrian crossing (for example: car, bus, truck, person, bicycle, motorcycle etc.)

- **Description of demonstration:** Deployed containers on the edge-computing box must be able to detect potentially dangerous situations, which in turn requires the ability to detect cars, trucks, buses, any vehicles approaching the pedestrian crossing from the road, as well as pedestrians walking in proximity of the crossing or directly on it.

- **Results:** The feasibility of the DECENTER-supported deployment underpinning this use case was fully demonstrated for the first pedestrian crossing located in the Trento neighbourhood of Vela. The results were illustrated during the second-year review. The experimentation allowed to assess the advantages from using the DECENTER technologies and highlight issues that need to be addressed when moving from a prototype to production-ready versions. The quality of images and video-capturing devices is key in minimising the interpretation errors
distinguishing between different types of moving vehicles (from a bus to a car to a motorcycle). Also, field validation on a real pedestrian crossing enabled to better assess the deployment and provide guidance for refinement of the decision-making AI services. The picture below illustrates all data directly collected or interpreted from the crossing and associated timing.

![Smart Crossing dashboard, showing identified vehicles/pedestrians, events and system reaction times.](image)

**KPI 2.** Ability to produce environmental data using IoT sensors deployed at the actual pedestrian crossing to identify different relevant and localized environmental conditions affecting road conditions.

- **Description of demonstration:** Additional sensors (small meteo-stations and sensing units for wet road conditions) were deployed for collecting environmental data and integrating IoT as additional data into the algorithms that control the crossing actuators.

- **Results:** real data from the pedestrian crossing experimentation has been collected and exposed through MQTT, providing an easy way to access the interface to manipulate the collected data at the edge. The demonstration showed the feasibility of such a location-based enhancement to the algorithms implemented to promptly control lights and buzzer notifications, instead of having to deal with latency and inaccuracy of backend cloud services and generic weather services.

**KPI 3.** Alert generation (performance of the system)

- **Description of demonstration:** this component in the demonstrator implemented the notification service for both vehicles on the road and pedestrians who are about to cross the road.

- **Results:** the results from using DECENTER can be seen in the picture below showing the latency between the time the data is collected from the street devices and the actuation being triggered as a result. The blue line highlights the case where all containers involved in the Alert
generation are deployed on the edge box as opposed to being all in a backend cloud (yellow line). The mixed scenario incurs additional delay due to the “to and for” communication between containers deployed in the edge and in the backend cloud.

4.2.3. Final Pilot Description

In the case of UC1 the decision has been to replicate in a second location what has already been successfully installed in the neighbourhood of Vela. The installation of a second pilot in a different location aims at testing the system in an environment that bears different characteristics (e.g. heavier car traffic, higher frequency of pedestrian traffic etc). The new features will therefore consist in validating the previous features in this new environment. At the time of writing this deliverable, the installation of the second pilot is work in progress.
4.3. Logistics Robotics Optimization

In Y2 the logistics robotics optimization was fully tested, verified and validated in a simulated environment as was described in D5.1. The main effort during Y3 has been dedicated to applying the same development to the real robots.

The following actions have been performed from the delivery of D5.1:

- Containerization in Docker containers of the ROS node modules specifics of the real robot. These modules require access to the real hardware to interact with it and have been divided as Figure 14 shows.

![Figure 14. Containerization of real robot modules classification.](image)

The Robot Core pod is in charge of interacting with at the hardware level with devices like the motor drivers, the battery charger, the pad controller, the battery monitor or the led manager. From the part of the sensors there is the Robot Sensors pod, that is who manages the containers to communicate to laser scan, to the RGBD camera and the inertial mass unit of the robot. They could not be tested in simulation so for this reason they were not integrated before.

- Integration with DECENTER Fog Platform. This allows placing robot resources across the available nodes regarding bandwidth and latency restrictions. Decenter Fog Platform has been integrated first on the simulation environment on the FBK premises where it was already deployed. This task comprehends the migration of the isolated Kubernetes manifest files into a new resource called model, that behaves like a cohesive group. This includes the simplification of the ROS services and fleet management system (FMS) ROS microservices.

- The integration with the Decenter Fog platform on the real robot environment was more complicated due to the fact that the devices where it was deployed are not server hardware. Creating the regions and relationships among them from scratch was required, and also deploying locally all needed manifests, and tuning some configuration to adapt to the new environment. Final modifications were also
performed to adapt the simulation model to the real environment, due intrinsic differences between them.

- Integration of Decenter SLA Manager.

The SLA Manager and SLA GUI have been integrated on the infrastructure provided by DECENTER where the robot environment is running. This task comprehends first the containerization of these two applications, and then the deployment and configuration of them in Kubernetes through YAML definition files.

The communication between the SLA Manager and the microservices is done through calls to the REST API exposed by the SLA Manager.

- Integration of Decenter LADS.

The L-ADS and Softflowd components have been integrated on the Robot environment infrastructure. These components have been containerized as Docker images, and then deployed in the Robot environment Kubernetes through YAML files.

The Softflowd component is responsible for gathering the information of the Robot microservices deployed on the same network. This information is sent later to the L-ADS application which is responsible for the ML processes.

- Integration of DECENTER microservices with the real robot.

Some of the DECENTER components as the Object Detector AI service, explained at section 3.2.3, need a direct connection with the camera feed. In this case the camera feed uses no compression format (RAW mode) so the robot microservices specifically created for DECENTER need a very stable and high bandwidth connection with the hardware.

In the simulation environment all nodes were connected with a wired high-speed connection, so this problem wasn't showing up. In the real environment the communication between the robot and the other devices is through a Wi-Fi network. This connection has lower speed and is less reliable than the Ethernet connection so delays and/or data lost can happen. In addition to this, the moving nature of the robot and electromagnetic interference made the connection not the most reliable link among cluster nodes.

In order to mitigate this problem, the detection microservice was forced to be placed on the robot. Figure 15 shows the different microservices through the Decenter Fog Platform interface.
Two real robots RB-1 Base (Figure 16) have been integrated into the demonstration and pilot scenario in order to justify the collaboration between robots thanks to the DECENTER platform integration. For instance, in the case where one robot detects the other in front of it and then decides to switch its path to avoid the other robot and achieve the goal. A deep description of the pilot is described in section 4.2.2.

The functionalities of picking and placing carts by robots have integrated into the demonstration. As a logistic operation the robots should be able to move material or goods from one point to another and this material is organized in carts. To achieve this objective, new modules related with the detection of the cart and the management of the elevator of the robot have been containerized and integrated into the demonstration. The Figure 17 shows two models of carts used during the demonstration tests.
The carts have some particularities to facilitate the pick and place operation by the robot autonomously. For instance, the functionality of the QR Code placed in front of the cart is for identifying the ID of the cart reading the code by the camera onboard the robot.

- A new set of images of rb1 robots have been used to train the DECENTER AI service and improve the results of the AI recognition microservice when a robot is identified.
- Several empiric tests in real environment have been performed inside Robotnik’s facilities in order to measure and demonstrate the KPIs and obtain the results shown in the next section.

### 4.3.1. MVD

The minimum Viable Demonstrator (MVD) of Y3 covers the following objectives:

- A. Vertical and horizontal offloading
- B. Picking and placing carts operation
- C. Safe human interaction

The MVD is built based on DECENTER features and the use case application requirements (Section 2.1.2 in D2.2 for a complete description). The following table lists the DECENTER features that are demonstrated and validated through this MVD in Y3:

- Security
- Computation offloading (horizontal and vertical)
- Digital Twin

In the following table we do the same with FR (functional requirement) by specifying which requirements have been realized at Y3:
<table>
<thead>
<tr>
<th>Functional Requirement</th>
<th>Description (&quot;The system must be able to...&quot;)</th>
<th>Comments on how to address the requirement</th>
<th>Y3 demo</th>
<th>Coverage Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_UC2_001</td>
<td>Trigger a sound or visual alarm</td>
<td>At least 1 speaker or led lights must be installed in robots</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC2_002</td>
<td>Allow the user to add missions</td>
<td>A graphical control interface is needed for human interaction</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC2_003</td>
<td>Allow the user to assign tasks to robots</td>
<td>A graphical control interface is needed for human interaction</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC2_004</td>
<td>Keep an updated map of the warehouse</td>
<td>A global knowledge about the environment at robots’ level. Communication between robots. Sensors to get information from the environment.</td>
<td>O</td>
<td>The Y3 demo works on an updated map of the environment integrated in Digital Twin (rviz)</td>
</tr>
<tr>
<td>FR_UC2_005</td>
<td>Detect persons</td>
<td>Camera and AI methods to detect humans in the surroundings of robots.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC2_006</td>
<td>Ensure the correct relation between providers through the use of smart contracts</td>
<td>The system will use safe and stables channels to use smart contracts</td>
<td>O</td>
<td>Integration of L-ADS carried out</td>
</tr>
<tr>
<td>FR_UC2_007</td>
<td>Get and show the status of robots</td>
<td>A graphical control interface is needed</td>
<td>O</td>
<td>The Digital Twin rviz is used to show the environment</td>
</tr>
<tr>
<td>FR_UC2_008</td>
<td>Keep track of tasks</td>
<td>A graphical control interface is needed</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC2_010</td>
<td>Provide statistical information</td>
<td>A graphical interface is needed to display the information</td>
<td>O</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 25. FR covered at Y3.

In the next table we do the same with NFR (non-functional requirement) by specifying which requirements have been realized:

<table>
<thead>
<tr>
<th>Non-functional Requirement</th>
<th>Description (&quot;The system must be able to...&quot;)</th>
<th>Comments on how to address the requirement</th>
<th>Y3 demo</th>
<th>Coverage Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR_UC2_001</td>
<td>Provide privacy of information (how many workers are working at each moment, number of boxes stored, ...)</td>
<td>Privacy of information</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>NFR_UC2_002</td>
<td>Delegate computational calculation to the Cloud</td>
<td>Vertical offloading</td>
<td>O</td>
<td>The demo Y3 includes vertical offloading demonstration and results</td>
</tr>
<tr>
<td>NFR_UC2_003</td>
<td>Delegate computational calculation to the Edge</td>
<td>Horizontal offloading AI algorithms running in the edge.</td>
<td>O</td>
<td>The demo Y3 includes horizontal offloading demonstration and results</td>
</tr>
</tbody>
</table>
D5.2 Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations

<table>
<thead>
<tr>
<th>NFR_UC2_004</th>
<th>Be reliable</th>
<th>Horizontal offloading</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR_UC2_005</td>
<td>Be safe for humans</td>
<td>Security sensors.</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 26. NFR covered by Y3.

In the following table are described the minimum viable demonstrator:

<table>
<thead>
<tr>
<th>Demo storyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start the actual robot fleet and AI application</td>
</tr>
<tr>
<td>Start a robot mission to pick a cart</td>
</tr>
<tr>
<td>Robot takes the shortest unblocked path and finish the mission</td>
</tr>
<tr>
<td>Add the same mission but with a person in front of the shortest path</td>
</tr>
<tr>
<td>When the robots detect an obstacle, send a picture to the AI</td>
</tr>
<tr>
<td>Fleet manager process the new information and update the mission</td>
</tr>
<tr>
<td>The robot which is detecting the person emits visual and sound alerts</td>
</tr>
<tr>
<td>When person moves, the robot follow and finish its mission</td>
</tr>
<tr>
<td>Add the same mission but with another robot stopped in front of the shortest path</td>
</tr>
<tr>
<td>When the robots detect an obstacle, send a picture to the AI</td>
</tr>
<tr>
<td>Fleet manager process the new information and update the mission</td>
</tr>
<tr>
<td>The robot with blocked path by another active robot changes the route</td>
</tr>
<tr>
<td>Add the same mission but with a box in the middle of the shortest path</td>
</tr>
<tr>
<td>When the robots detect an obstacle, send a picture to the AI</td>
</tr>
<tr>
<td>Fleet manager process the new information and update the mission</td>
</tr>
<tr>
<td>The robot with blocked path by a box changes de route</td>
</tr>
</tbody>
</table>

4.3.2. UC2 KPIs

KPI 1. Precision of the identification of the obstacle detected

- Description of demonstration: When the robot detects some obstacle on its way it calls the AI service of DECENTER and within the answer the AI service returns a confidence value of the identification of the object.

The possible object tags to recognize are Person and/or Robot. Anything else will be identified as “other”. That means that a box will be recognized as “other”. Also means that a recognition of a robot or a person with a confidence below a certain threshold will be tagged as “other” thing.

This confidence value depends on several factors like the robot position, the object position, the ambient light, the reflection of the object, etc. To measure this KPI, a set of fifty tests of each type (robot, person and other) have been performed.
For robot and person, the test consists of placing an obstacle in front of the robot, calling the AI service and taking note of the confidence value returned.

For the “other” tag it will measure if there is a wrong tagging or false positive.

- **Results:** A set of hundred and fifty tests has been performed placing random objects in front of the robot in order to force it to call the AI service and check the confidence value returned. The results have been divided into the three cases that the AI service can identify: person, robot or something else (like a box). The Figure 18 shows an example of these three cases.

![Figure 18. From left to right: person, robot, wooden box.](image)

During these tests, 50 objects placed in front of the robot were human, 50 objects were robots and 50 were something else (usually just a wood box). Figure 19 shows an analysis of the 50 tests where a human was in front of the robot.

![Figure 19. Person detection results.](image)

The Figure 19 shows the results of the AI Object recognition service when a person is in front of it. On the graph it is possible to extract that on most of the test samples the confidence goes from around 86% to 98% resulting in a mean value of 90.27%. However, some of the samples,
in particular two of them, have low confidence. That could be because of low light, backlight conditions or blurry pictures.

A threshold has been defined to ensure the correct identification so when the confidence value is lower than 75%, the output of the identification is “other thing” to not compromise the result, so in these two cases with low confidence the return value by AI Object recognition was mistakenly “other”.

The same was done for 50 tests with a robot in front.

![Graph showing AI Robot Recognition Confidence Level](image)

*Figure 20. 50 samples of the AI Object recognition service when a robot is in front of it.*

On the graph of Figure 20 it is possible to extract that on most of the test samples the confidence goes from around 87% to 96% resulting in a mean value of 90.33%. But again, some of the samples, three in particular, have a low confidence because of light conditions, strange perspective, or blurry pictures. Therefore, the outcome of those samples was wrongly identified as “other”.

For the kind “other” the algorithm does not return any confidence value and during the 50 tests done with the wooden box placed in front of the robot, the algorithm always returned “other” so for the next analysis only the tests done with persons and robots are taken into account.

The Figure 21 shows the combination of the 100 test results of the person and robot detection. The dots in blue are the person samples and the orange ones are the robot samples. The five isolated samples with a quite low confidence value (below 75%) were tagged as “other” so the error in this case results a 5%.
In conclusion, the results of the testing for this KPI conclude that the main confidence level is higher than 90.30% for detection of the robot or person and the correct tagging of other is above 91%. The objective at the end of the project was to reach a confidence value up to 90%, so the objective has been successfully accomplished.

KPI 2. Accuracy of the estimation position of the obstacle detected.

- **Description of demonstration:** The position estimation of the obstacle detected is measured with a laser scan sensor. The laser scan installed in the robot is a Hokuyo UST-10LX model, it covers 270° the perimeter of the robot with a range of 10m and a precision of +/−40mm. The Figure 22 shows the workspace of the laser scan around the robot, drawing the data returned by the laser in red.
The position estimation of the object in front of the robot is crucial to stop the robot before a possible collision and also to know if the object is in the path of the robot or not. To compare the measure calculated by the system and the real one, 20 tests have been performed. Random objects were placed in front of the robot at different distances. Per each test the value returned by the estimation position algorithm is compared with the real measure taken by hand.

The Figure 23 shows an example of one test where the distance of the box has been measured by hand with a measuring tape in order to be compared with the estimated position by the algorithm.
- **Results:** The Figure 24 shows the results of the 70 tests where an object was placed in front of the robot at 7 different known distances. The position estimation of the object (in red dots) is compared with the actual distance measured by hand (in blue lines).

![Figure 24. Comparison between real distances measured by hand and distances calculated by robot.](image)

The error understood as the difference between the measures reaches 5 cm in the worst cases. If the average is calculated based on the distance, the results are the ones that Figure 25 shows.

![Figure 25. Distance average measurement error.](image)
The maximum average value of the error is 0.6 cm in the worst case but the total average error is 0.02 cm. The standard deviation error per each distance has been also plotted at Figure 26.

![Figure 26. Standard deviation error of the distance measures.](image)

In conclusion, the standard deviation error resulting is 1.57 cm so it can be concluded that the measurement estimation is below 2 cm as was dictated for this KPI.

**KPI 3. Saving CPU robot consuming**

- **Description of demonstration:** The idea to demonstrate this KPI is to compare the CPU consumption when the robot is running the metal bare software (before DECENTER) with the DECENTER containerized version of software. This last version allows the PODs movement (with horizontal/vertical offloading) to other external devices or to the robot-like edge devices so the computation is shared between them decreasing the computation consumption of the robots and saving energy.

- **Results:** The results have been analysed in two different cases: one when the vertical offloading is performed and another one when horizontal offloading happens.

**Vertical Offloading**

The vertical offloading is used to delegate computation into the edge in order to unload the robot CPU saturation.

In this test three regions were involved: robot 1, robot 2 and the edge device. At the beginning of the test, robot 1 and 2 have a similar computation consumption due to the fact that they are executing the PODs needed to perform the autonomous logistics operations. The average in mCPU for each one is around 2000 mCPU. On the other hand, the edge device (an Intel NUC minipc) is almost idle, executing just the minimal computation. This situation is reflected during the first 45 seconds of Figure 27.
Then, at time 45 sec, the vertical offloading is performed and some PODs from robot 1 are delegated into the edge region device. The process of changing the context unloading and uploading the PODs into the regions generates some brief overload as is shown in the different graphs of Figure 27. After that moment, the CPU consumption of robot 1 decreases almost to the half and the CPU consumption of the edge device increases almost until the double. Concretely, The CPU load average of the edge region before performing the offloading was 625mCPU and after the offloading average increased to 1963mCPU. At the same time, the CPU load average of the robot 1 region before performing the offloading was 1963mCPU and after the offloading average decreased to 1340mCPU. And obviously, for the case of the CPU load average of the robot 2 region, both before and after the performance of the offloading, it was 1996mCPU.

With the objective to analyse the computation consumption of the whole system, the Figure 28 reflects the total cluster CPU usage in all regions.

**Figure 27. CPU. Consumption of involved devices during the vertical offloading test.**
D5.2 Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations

Figure 28. Total CPU usage of the whole cluster during the vertical offloading test.

The figure shows the accentuated overload that produces make the vertical offloading at sec 45 but apart from that the total CPU usage is constant during the experiment even when the CPU consumption of Robot 1 decreases and the CPU consumption of Edge increases.

In conclusion, the results show that with the vertical offloading the CPU saturation on the robot 1 was reduced around 31%, overpassing the goal dictated by the KPI that was set in at least 10%.

Horizontal Offloading

The horizontal offloading is similar to the vertical offloading but in this case an auxiliary robot is used as the selected device to unload the main robot CPU saturation. As it was done in the previous tests, the measures of CPU consumption of the three devices involved in the experiments have been plotted in Figure 29.

Figure 29. CPU consumption during the horizontal offloading.

At the beginning of the test, Robot 1 was consuming an average of 1963mCPU and Robot 2 around 750mCPU because it was not navigating autonomously or was in an idle state. At 1 hour of the experiment, the horizontal offloading is performed and the load from Robot 1 is unloaded and delegated to Robot 2. The graphs show the exact moment when this happens. Then, the CPU consumption of Robot 1 decreases to 1340mCPU and the one of Robot 2 increases until around 1750mCPU on average. In this test the region of edge is part also of the cluster but it has no movements of load so the CPU consumption was constant during the whole experiment.

The Figure 30 shows the CPU consumption of the complete cluster during all the experiments. The overload appears when the offloading starts as it also happens with the vertical offloading
and it is due to the process of switching off or unloading the pods involved and starting the new ones in the remote region.

![CPU consumption - Cluster](image)

*Figure 30. Cluster CPU consumption during the horizontal offloading.*

In conclusion, the results show that with horizontal offloading the CPU saturation on the robot 1 was reduced around 32%, so the KPI of reducing the CPU saturation by 10% is successfully achieved in either vertical and horizontal offloading.

### 4.3.3. Final Pilot Description

During Y2 all containers involved in the pilot as the fleet management system or the AI microservices were developed, tested and verified on a simulated environment based on Gazebo simulator, as it was described at D5.1.

The definition of the pilot has not changed from the delivering of D5.1. The history behind the pilot is the same, the main difference is that everything runs within the real hardware, the autonomous robots, instead of a simulation environment.

The final pilot was performed inside Robotnik’s facilities. As it was described in the simulation environment, the pilot room had two doors to go inside it and the initial position of the robot was out of the room as the next figure represents.

![Scenario Representation](image)

*Figure 31. Scenario Representation (left). Real scenario (right).*
There is a door (door A) that is closer to the goal that the robot has to reach so preferably, the path planning microservice always chooses that way to enter into the room (green path drawn at the images).

During the pilot, there were three different types of obstacles placed into the door A: a wooden box, a person and another robot. The wooden box represents any static obstacle that is not a person or a robot, so for the AI microservices the results of its classification is “others”.

Without DECENTER, the robot was not able to identify which kind of object was in front of it so the usual behaviour is just to stop and wait some time or wait until the obstacle disappears.

With DECENTER, each time an obstacle is found in front of the robot, the AI service is called to identify the object and actuate properly. The Figure 32 shows the case of the wooden box.

**Figure 32. Scenario where there is a box in front of the robot.**

When the robot is in front of the box it takes a picture and sends it to the AI microservice. The picture and the result can be shown below.

```json
INFO:__main__:Return message: {
  "id": "d87b545d-98d7-4524-9662-1ca63b25a715",
  "img_url": "http://object-identifier:5001/static/d87b545d-98d7-4524-9662-1ca63b25a715.jpg",
  "output_img_url": "http://object-detector-yolov3:5000/static/d87b545d-98d7-4524-9662-1ca63b25a715_output.jpg",
  "request": {
    "confidence_limit": 0.4,
    "file": " rb1_base_collision_image.jpg",
    "img_timestamp": "12-02-2021T09:29:47.395387",
    "robot_pose_theta": -0.371941073781,
    "robot_pose_x": 5.70204481131,
    "robot_pose_y": 2.08444489208
  },
  "result": [{
    "detection_details": {},
    "warehouse_obj_class": "others"
  }],
  "status": "success",
  "timestamp": "12-02-2021T09:29:48.355627"
}
```

**Figure 33. Outcome of AI microservice (left). Picture sent by robot (right).**
In case of the identification of the wooden box as “others”, the system assumes that the path of door A will be blocked forever so it needs to replan sending the robot through the long path crossing the other door (door B).

The second case happens when a person is standing in front of the robot as Figure 34 shows.

![Figure 34. Scenario where a person is in front of the robot.](image)

In this case the flow is similar. When the robot detects the obstacle, it stops, it sends the picture to the AI microservice and the result is the following:

```json
INFO:__main__:Return message: {
    "aipid": "53ff29dd-2ace-4648-b4f7-733611c4363f",
    "request": {
        "confidence_limit": 0.4,
        "file": "rb1_base_collision_image.jpg",
        "img_timestamp": "2021-02-12T10:14:15.774204",
        "robot_id": "0",
        "robot_pose_theta": -0.638679469787,
        "robot_pose_x": 5.89503095742,
        "robot_pose_y": 1.40157587692
    },
    "result": {
        "detection_details": {
            "confidence": 0.9416,
            "coordinates": {
                "xmax": 395,
                "xmin": 213,
                "ymax": 316,
                "ymin": 85
            },
            "obj_class": "person"
        },
        "warehouse_obj_class": "person"
    },
    "status": "success",
    "timestamp": "2021-02-12T10:14:17.154382"
}
```

![Figure 35. “Person” output from AI microservice (left). Picture of person sent by robot (right).](image)
When a person is detected the behaviour of the robot is different. The robot starts blinking its lights and making noise for 5 seconds to advertise the person that he is on the path of the robot. Once the person moves, the robot starts moving again through the shortest path (door A) and reaches its goal.

The last case focuses on the scenario where an autonomous fleet of robots is managing a warehouse. In this case robots move around sharing their space and their routes with the others. In these situations, the deadlock between robots is quite common in narrow spaces where one robot is trying to exit from one place and another is trying to get into it (like doors, corridors, lifts...).

Like in the other cases, the robot stops when it detects the obstacle on its path, it takes a picture and it sends it to the AI microservice. The result is the following:

```
INFO:__main__:Return message: {
  "aipid": "69be3852-4282-469a-8ded-5c7d7e961d",
  "img_url": "http://object-identifier:5001/static/69be3852-4282-469a-8ded-5c7d7e961d.jpg",
  "output_img_url": "http://object-detector-yolov3:5000/static/69be3852-4282-469a-8ded-5c7d7e961d_output.jpg",
  "request": {
    "confidence_limit": 0.4,
    "file": "rb1_base_collision_image.jpg",
    "img_timestamp": "12-02-2021T09:54:06.253388",
    "robot_id": 0,
    "robot_pose_theta": -0.584536687593,
    "robot_pose_x": 6.01231350625,
    "robot_pose_y": 1.37102538138
  },
  "result": [
    {
      "detection_details": {
        "confidence": 0.8917,
        "coordinates": {
          "xmax": 628,
          "xmin": 182,
          "ymax": 487,
          "ymin": -22
        },
        "obj_class": "robot"
      },
      "warehouse_obj_class": "$\text{robot}$
    }
  ],
  "status": "success",
  "timestamp": "12-02-2021T09:54:07.520903"
}
```

When the outcome of the AI microservice is from the kind of “robot", the system replans the route to continue through the longest path (door B) but the closest path will not be considered blocked permanently because it will be free again when the other robot moves.

All cases can be seen in detail at the final video of the pilot.

### 4.4. Smart and Safe Construction

The Smart and Safe Construction use case addresses the requirements of the dynamic construction process, such as: constant information support and high QoS. Due to the dynamic nature of the construction process, organising, monitoring, and implementing a construction project whilst achieving high standards of safety, security, logistics, inspection, and other aspects can be very challenging. Hence, UC-3 addressed the requirement of improving the safety and security at construction sites. UC-3 implemented the AI models developed in WP4 to perform object detection and member verification from live video streams; integrated with the orchestrator from WP3 to perform QoS-aware orchestration of AI models in the Fog; integrated with the cross-border data management services from WP4 for data management and to satisfy privacy regulations.
Due to the COVID-19 limitations and the dynamic nature of the construction site, resulting with a completely different scene from the time of development to the time of final demonstration, the intended scenario will be demonstrated by using pre-recorded video from a construction site in Slovenia.

4.4.1. MVD

The Minimum Viable Demonstrator (MVD) of Y3 covers the following objectives:

A. QoS-aware resource orchestration
B. Trustworthy and secure cross-border data management
C. Seamless selection from a variety of AI models

The MVD is built based on DECENTER features and the use case application requirements (Section 2.1.3 in D2.2 for a complete description). The following table lists the DECENTER features that are demonstrated and validated through this MVD in Y3:

- QoS-aware resource orchestration.
- AI Model Repository and model management.
- DECENTER AI package and AI models.
- Blockchain-based data management.

In the following table we do the same with FR (functional requirement) by specifying which requirements have been realized at Y3:

<table>
<thead>
<tr>
<th>Functional Requirement</th>
<th>Description (&quot;The system must be able to...&quot;)</th>
<th>Comments on how to address the requirement</th>
<th>Y3 demo</th>
<th>Coverage Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_UC3_001</td>
<td>Access and use existing pre-trained AI database models</td>
<td>Certain pre-trained AI models already exist, thus it is possible to retrieve them from a database/repository and use them. Currently the DECENTER AI Model Repository is being used.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC3_002</td>
<td>Train and additionally customize existing AI models (transfer learning)</td>
<td>Pre-trained model for identification of specific objects such as trucks already exists (e.g. AlexNet or YOLO), which are being used. It would be also possible to additionally customize the model, for example, to identify helmet in addition to a hat.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC3_003</td>
<td>Receive and process data from a video camera</td>
<td>Receive and ingest a video stream, and extract individual frames for processing and analysis</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC3_004</td>
<td>Place bounding boxes at specific (interesting) images parts (object detection)</td>
<td>A specific algorithm for image segmentation is used, which results in specific bounding boxes of image segments.</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>FR_UC3_005</td>
<td>Trigger notifications for construction site engineer</td>
<td>All features identified will be sent as notifications to the mobile phone of the construction site manager if certain conditions are met. (Rule based system for notifications will be implemented).</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
</tbody>
</table>
In the next table we do the same with NFR (non-functional requirement) by specifying which requirements have been realized:

<table>
<thead>
<tr>
<th>Non-functional Requirement</th>
<th>Description (“The system must be able to…”)</th>
<th>Comments on how to address the requirement</th>
<th>Y3 demo</th>
<th>Coverage Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR_UC3_001</td>
<td>The smart application shall be able to use more or less video cameras and shall be reused in different layouts with respect to different construction sites.</td>
<td>Scaling the number of available Fog nodes by using Smart Contracts to adapt for increasing number of video cameras.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>NFR_UC3_002</td>
<td>Keep private the information processed on each construction site.</td>
<td>Smart Contracts and Oracles are specified for individual Fog Nodes to facilitate access to sensitive information and control the access.</td>
<td>O</td>
<td>Covered at Y2. Smart Oracles to be presented in Y3.</td>
</tr>
<tr>
<td>NFR_UC3_003</td>
<td>Provide a processing time of object detection 30 seconds.</td>
<td>System must promptly detect a violation. For instance, if a person is wearing a helmet before the person can reach restricted area from the entrance or the office of the construction site; in a distance of at least 4.0 m (the width of manipulation intervention road) with a presumption that average walking speed is 1.4m/s.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>NFR_UC3_004</td>
<td>Have the possibility to operate even if specific Fog Node fails to respond on time.</td>
<td>The system will use Smart Contracts to find out additional Fog Nodes that can be used for the process.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>NFR_UC3_005</td>
<td>Perform correctly in different temperature/illumination conditions</td>
<td>This shall be achieved through proper set up of the sensors and video cameras.</td>
<td>O</td>
<td>-</td>
</tr>
</tbody>
</table>

In the following table are described the minimum viable demonstrator:

<table>
<thead>
<tr>
<th>Demo storyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose video stream</td>
</tr>
<tr>
<td>Choose AI method</td>
</tr>
<tr>
<td>Choose region with Fog Nodes</td>
</tr>
<tr>
<td>Define QoS requirements</td>
</tr>
<tr>
<td>Initiate deployment</td>
</tr>
<tr>
<td>Background deployment activities: (1) optimal Fog Node is determined, (2) Smart Contract is executed and transaction confirmed, (3) AI method is deployed.</td>
</tr>
<tr>
<td>On a new screen, the video stream is shown alongside with output of the AI model that constantly appears.</td>
</tr>
</tbody>
</table>
A violation happens and a notification is sent to user.

When the user requires more time for the AI model to process data, a new transaction is executed with the Smart Contract.

When the given time passes or the user stops the model, there is a notification that the model has stopped working and the initial dashboard is shown.

When another user, that has no permission tries to run the AI model, the Smart Contract rejects the transaction and the deployment is not fulfilled.

4.4.2. UC3 KPIs

KPI 1. AI model response time performance: This is a quantitative KPI, that measures the response time in seconds, between a safety violation event taking place (i.e., object detection/member verification) and the moment of receiving the notification.

- Description of demonstration: For the purposes of this KPI, the AI models was 20 times deployed with and without using the QoS-aware orchestration algorithms of the DECENTER Platform. Then for each case, the time between a safety violation event (i.e., object detection/member verification) and the received notification, was estimated. The goal of this KPI is to demonstrate the improved QoS (i.e., better response time) when utilising the DECENTER Platform in the context of Smart and Safe construction scenarios.

- Results: The monitoring results were collected in a form of system logs with events' timestamps, which were later inspected. The results presented in Figure 37, show significant improvement in the performance, when the DECENTER Platform is used. Average response time without QoS aware orchestration is above 150 ms, whereas the average response time with QoS aware orchestration is less than 60 ms.

![Figure 37. AI model response time performance (KPI1).](image-url)
KPI 2. Temporary use of external deployment options: This is a quantitative KPI, that counts the number of available providers in the DECENTER Fog and Brokerage Platform. The KPI estimates the operational cost reduction in infrastructures (i.e., cost of ownership vs. cost of temporary leasing).

- Description of demonstration: Economic estimation of the operational cost reduction in infrastructures: cost of ownership vs. cost of temporarily leasing

- Results: To estimate the cost of owning and leasing the computing nodes, we considered the expenses that a construction company in Slovenia (e.g., Šumi d.o.o.) would have. For the purposes of the estimation, it was considered that the construction site operates 20 days per month, while the computing infrastructures will be required 12 hours per day. Hence, the infrastructure would be operating 2880 hours per year. The table below provides the temporary leasing prices for the computing nodes where the company had to run the AI models. According to the estimates of the company, to own the computing infrastructures would be much more expensive (i.e., they would have to pay more than 20000 € per year), due to additional costs, such as: employment of additional stuff, electricity, and air-conditioning costs, hardware failures etc. Following their estimations, using the DECENTER Fog and Brokerage Platform, where large quantity of nodes is offered, in average can reduce their yearly expenses up to 90%.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Type (GPU instances)</th>
<th>Cost [€/year]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arctur</td>
<td>HPC GPU node</td>
<td>538.56</td>
</tr>
<tr>
<td>Amazon AWS</td>
<td>g4dn.xlarge</td>
<td>1786.6</td>
</tr>
<tr>
<td></td>
<td>g4dn.2xlarge</td>
<td>2534.4</td>
</tr>
<tr>
<td></td>
<td>g4dn.4xlarge</td>
<td>4060.8</td>
</tr>
<tr>
<td>Google GCP</td>
<td>NVIDIA T4</td>
<td>633.6</td>
</tr>
<tr>
<td></td>
<td>NVIDIA P4</td>
<td>1177.92</td>
</tr>
</tbody>
</table>

Table 29. Estimation cost of service providers.

KPI 3. Improve intelligence of smart construction site using on-demand AI models: This is a quantitative KPI that counts the number of AI models that are ready to be employed at a construction site and compares the time required to deploy a new AI model with and without DECENTER.

- Description of demonstration: Demonstrate the number of available AI models from the AI model repository and the time to deploy a new AI model.

- Results: The results are presented in the table below.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Before DECENTER</th>
<th>After DECENTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of available and pretrained AI-models</td>
<td>N/A</td>
<td>3</td>
</tr>
<tr>
<td>for construction sites</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time necessary to run AI models</td>
<td>1-5 days. No AI model repositories</td>
<td>10-30 minutes. Due to the A</td>
</tr>
</tbody>
</table>
new AI model were previously implemented. Each AI model had to be trained, adapted, and configured from scratch and for each construction site separately. A model repository that contains pretrained AI models, and the DECENTER AI Package that offers pre-set means of communication, implementing new AI models has become much simpler and faster operation.

Table 30. KPIs before and after DECENTER.

KPI 4. Improve data privacy, safety, and security in smart construction sites: This is a qualitative KPI that leverages blockchain technologies for trustworthy interactions with AI models.

- **Description of demonstration:** Demonstrate that AI models can run only upon executing a Smart Contract that is designed to satisfy privacy regulation, whilst keeping low cost of operation. In order to successfully execute the Smart Contract function, it has to fetch trustworthy data (i.e. access information) through a Smart Oracle.

- **Results:** Before DECENTER there was no information of such implementation in practice. After DECENTER, Smart Contracts and Oracles used to facilitate trustworthy access to AI models and achieve transparency and traceability when using the AI models. In our experimental environment we used the Ethereum testing environment Rinkeby and we presented the averaged results of 10 executions, running own Chainlink nodes with the determined Smart Oracle cost per interaction of 0.001 LINK tokens, which is equivalent to the main Ethereum environment of approximate 0.037 USD at the time of experimentation.

4.4.3. **Final Pilot Description**

In the final version of UC-3, all microservices (see Figure 17) comprising the AI application have been developed, tested, and deployed. In particular: (1) The necessary AI models (i.e. object detection and person detection/identification) have been implemented, (2) AI models have been optimized and improved, (3) developed and implemented smart contracts to facilitate access to AI models, (4) developed and implemented smart oracles to facilitate trustworthy off-chain data management, (5) develop and implement the monitoring and orchestration components, (6) Integrate with DECENTER Platform and services (i.e. Resource orchestration, DECENTER AI Package, DECENTER Data Management).

The source code for those microservices can be found in DECENTER project’s GitLab repository. CI/CD (continuous integration/continuous delivery) pipelines are already running to integrate, package and publish those microservices as Docker containers (in a docker images repository in the same GitLab system).

The final demonstration plan is composed of 7 consecutive steps that will demonstrate the operability of all software components:
(1) The user (i.e. construction manager) uses DECENTER User Interface to select AI methods (i.e. object detection, member verification) and a video streaming source from the available cameras at the construction site; The user also defines the QoS requirements (i.e. SLOs) and agrees upon the SLA agreement

(2) The Trusted Model Manager (TMM) requests permission via the Blockchain Service (BS) to pull the chosen AI method.

(3) BS triggers the execution of Smart Contract (SC), which communicates with Smart Oracles (SO) to verify AI method access regulations and policies.

(4) When the SC executes successfully, TMM requests the AI method from Data/Access Manager. Data/Access Manager verifies the BC transaction. In case the verification is successful, Data/Access Manager grants permission to pull the model from the DECENTER AI repository.

(5) DECENTER Fog Platform determines an optimal deployment option where the AI method will run.

(6) The container running the AI model starts to receive video stream data from the camera, process the data and send context results to the Message Broker.

(7) Once a violation is detected (e.g., unauthorized personnel/object on premises), the construction manager receives a notification.

4.5. Ambient Intelligence for Office Environments

Third year, all the device and platform had been installed on the KETI's laboratory in Seoul, Korea. We also designed new scenario to show the advantage of DECENTER in the viewpoint of AI model substitution. The time to change the AI model of microservices can be shortened through the methods provided by the DECENTER platform. Tests to achieve KPIs were completed by April 2021.

4.5.1. MVD

The minimum Viable Demonstrator (MVD) of Y3 covers the following objectives:

A. Face detection and feature extraction
B. Member verification
C. AI Model substitution

The MVD is built based on DECENTER features and the use case application requirements (Section 2.1.2 in D2.2 for a complete description). The following table lists the DECENTER features that are demonstrated and validated through this MVD in Y3:

- AI model substitution
- Face detection and member verification accuracy evaluation
- AI model repository and Resource management connection
In the following table we do the same with FR (functional requirement) by specifying which requirements have been realized at Y3:

<table>
<thead>
<tr>
<th>Functional Requirement</th>
<th>Description (“The system must be able to...”)</th>
<th>Comments on how to address the requirement</th>
<th>Y3 demo</th>
<th>Coverage Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_UC4_001</td>
<td>Extract features from images and detect moving objects</td>
<td>Edge will have an object detector, which is able to produce a feature map as an input of another engine.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC4_002</td>
<td>Have the ability to change video data input to image format and change the image data input to an interpretable size and shape by AI engine</td>
<td>The system will have an image preprocessor based on use-case scenario.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC4_003</td>
<td>Check whether the detected human has entered the restricted area</td>
<td>The system has a human detector which can detect the access of people in a specific space.</td>
<td>O</td>
<td>UC4 Scenario was changed. Face detector detects face when anybody visits a specific space.</td>
</tr>
<tr>
<td>FR_UC4_004</td>
<td>Detect whether a visitor is an authorized member or not</td>
<td>The system has a member verifier based on registered images.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC4_005</td>
<td>Detect a face in the given input</td>
<td>The system has a face detector which detects the location of a face from the given input.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>FR_UC4_007</td>
<td>Store IoT sensor data (e.g., PM10) based on specific use-case scenario</td>
<td>Data storage will retain data for a specific period of time based on use-case scenario.</td>
<td>O</td>
<td>Covered at Y3</td>
</tr>
<tr>
<td>FR_UC4_008</td>
<td>Trigger alerts when a potentially dangerous situation is perceived or an error on a task occurs</td>
<td>The system will be connected to the service application which can trigger alerts.</td>
<td>O</td>
<td>Covered at Y2, Scenario was changed, it provides important information according to the result of member verification</td>
</tr>
</tbody>
</table>

Table 31. FR coverage.
In the next table we do the same with NFR (non-functional requirement) by specifying which requirements have been realized:

<table>
<thead>
<tr>
<th>Functional Requirement</th>
<th>Description (“The system must be able to…”)</th>
<th>Comments on how to address the requirement</th>
<th>Y3 demo</th>
<th>Coverage Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR_UC4_00_1</td>
<td>Have a reliable member verification</td>
<td>The accuracy of member verification will be evaluated on face retrieval protocol, and the accuracy will be at least 70%.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>NFR_UC4_00_2</td>
<td>Have one or more predictors that can infer the indoor future conditions</td>
<td>This system will predict more than two indoor future conditions.</td>
<td>O</td>
<td>UC4 Scenario was changed.</td>
</tr>
<tr>
<td>NFR_UC4_00_3</td>
<td>Have a method for evaluating the performance of the human detector</td>
<td>The performance of the human detector will be evaluated by drawing precision-recall curve on public benchmark.</td>
<td>O</td>
<td>UC4 Scenario was changed. We detect human face and group name.</td>
</tr>
<tr>
<td>NFR_UC4_00_4</td>
<td>Get the face detection result during a proper time</td>
<td>The time response is highly dependent on the capabilities of each edge. The system will respond in less than 1 minute for the service. This system does not deploy AI solutions for some specific edge devices that are expected to take a longer time.</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
<tr>
<td>NFR_UC4_00_5</td>
<td>Select the appropriate model for a specific scenario that will run at the edge</td>
<td>In the case of indoor environment prediction, either cloud or edge will be able to select</td>
<td>O</td>
<td>Covered at Y2</td>
</tr>
</tbody>
</table>
In the following table are described the minimum viable demonstrator:

<table>
<thead>
<tr>
<th>Demo storyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start</strong></td>
</tr>
<tr>
<td>Deploy microservices to edge resources using DECENTER Fog Platform</td>
</tr>
<tr>
<td>UC4 GUI to validate deployment of microservices</td>
</tr>
<tr>
<td>A person approach to the camera</td>
</tr>
<tr>
<td>UC4 GUI to validate data flow between microservices</td>
</tr>
<tr>
<td>Display device shows meeting room information according to the person's information detected</td>
</tr>
<tr>
<td>Un-deploy all the microservices.</td>
</tr>
<tr>
<td>Deploy microservices, with FD on embedded GPU</td>
</tr>
<tr>
<td>Validate performance metric with UC4 GUI</td>
</tr>
<tr>
<td>Un-deploy and deploy microservices, with MV on cloud</td>
</tr>
<tr>
<td>Validate performance metric with UC4 GUI</td>
</tr>
</tbody>
</table>

| NFR_UC4_006 | Transfer the result of the AI engine from one edge to other edge | Each AI engine will connect to another engine through a local network. | O | Covered at Y2 |

*Table 32. NFR coverage.*
4.5.2. KPIs

a. DT and SL KPIs

All KPIs described in Section 4.1 (DT1-DT2 and SL1-SL5) were implemented and demonstrated in the context of UC4. The table below presents the results of the DT KPIs.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT1: Assets Integration</td>
<td>1 AI (face detection), 1 IoT device (camera), 8 Digital Entities (e.g., Person, Group, Location)</td>
</tr>
<tr>
<td></td>
<td>1 communication protocol: MQTT</td>
</tr>
<tr>
<td>DT2: Data Integration</td>
<td>Not applicable. There is not a constant flow of data, only periodic data when a user stands in front of a camera</td>
</tr>
</tbody>
</table>

Table 33. Results of DT KPIs.

The results of the semantic layer KPIs are also presented for UC4, along with a brief description of the methodology followed in order to obtain them.

KPI SL1: Completeness

Description of demonstration: The functional requirements of UC4 (described in D2.2) were contrasted with the entities in the ontology of the semantic layer. The table below illustrates the completeness of the ontology, i.e. whether and how each requirement is addressed by the semantic layer. Functional requirements that are not relevant with data representation (e.g., change video to image) are omitted.

<table>
<thead>
<tr>
<th>FR ID</th>
<th>FR description</th>
<th>Semantic representation</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_UC4_003</td>
<td>Check whether the detected human has entered the restricted area</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#Person">www.semanticweb.org/ontologies/2021/sensiNactOntology#Person</a>,</td>
<td>sensiNactOntology</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#hasRelativeLocation">www.semanticweb.org/ontologies/2021/sensiNactOntology#hasRelativeLocation</a>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#Location">www.semanticweb.org/ontologies/2021/sensiNactOntology#Location</a></td>
<td></td>
</tr>
<tr>
<td>FR_UC4_004</td>
<td>Detect whether a visitor is an authorized member or not</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#Person">www.semanticweb.org/ontologies/2021/sensiNactOntology#Person</a>,</td>
<td>sensiNactOntology</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#belongs">www.semanticweb.org/ontologies/2021/sensiNactOntology#belongs</a>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#Group">www.semanticweb.org/ontologies/2021/sensiNactOntology#Group</a></td>
<td></td>
</tr>
<tr>
<td>FR_UC4_006</td>
<td>Predict the future value of environmental factors</td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#temperature">www.semanticweb.org/ontologies/2021/sensiNactOntology#temperature</a>,</td>
<td>sensiNactOntology</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#humidity">www.semanticweb.org/ontologies/2021/sensiNactOntology#humidity</a>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#rain">www.semanticweb.org/ontologies/2021/sensiNactOntology#rain</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#luminance">www.semanticweb.org/ontologies/2021/sensiNactOntology#luminance</a>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.semanticweb.org/ontologies/2021/sensiNactOntology#wind">www.semanticweb.org/ontologies/2021/sensiNactOntology#wind</a></td>
<td></td>
</tr>
</tbody>
</table>
FR_UC4_007 Store IoT sensor data (e.g., PM10) based on specific use-case scenario
www.semanticweb.org/ontologies/2021/sensiNactOntology#Provider,
www.semanticweb.org/ontologies/2021/sensiNactOntology#provides,
www.semanticweb.org/ontologies/2021/sensiNactOntology#Service /
www.semanticweb.org/ontologies/2021/sensiNactOntology#Service,
www.semanticweb.org/ontologies/2021/sensiNactOntology#hasResources,
www.semanticweb.org/ontologies/2021/sensiNactOntology#Resource

FR_UC4_008 Trigger alerts when a potentially dangerous situation is perceived or an error on a task occurs
www.semanticweb.org/ontologies/2021/smartCrossingOntology#DangerousSituation,
instanceof,
www.semanticweb.org/ontologies/2021/smartCrossingOntology#Event

| Table 34. FR contrasted with existing entities in the ontology |

Results

The table below summarizes the percentage of functional requirements being covered by the ontologies and how much of these requirements are being covered by each ontology (sensiNactOntology, SmartCrossing). The zero coverage of SmartCrossing ontology is expected, since it is a dedicated ontology designed for UC1.

<table>
<thead>
<tr>
<th>DT1: Completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
</tr>
<tr>
<td>UC4</td>
</tr>
</tbody>
</table>

| Table 35. Completeness of SL |

KPI SL2: Conciseness

Description of demonstration: The table below shows which of the ontological entities are related with UC4 domain. The higher the number of entities in an ontology being related to a domain, the more precise the ontology is considered to that domain.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Included in UC4</th>
<th>sNaOntology</th>
<th>SmartCrossing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td>x</td>
<td></td>
<td>Pedestrian</td>
</tr>
<tr>
<td>Service</td>
<td>x</td>
<td></td>
<td>Cyclist</td>
</tr>
<tr>
<td>Resource</td>
<td>x</td>
<td></td>
<td>Stroller</td>
</tr>
<tr>
<td>Person</td>
<td>x</td>
<td></td>
<td>Petowner</td>
</tr>
</tbody>
</table>
The percentage of ontology’s entities also included in the UC4 represents the conciseness of the semantic layer. The table below shows the overall conciseness of the semantic layer, as well as the percentage of relevant entities to UC4 for each ontology. The zero conciseness of SmartCrossing ontology is expected, since it is a dedicated ontology designed for UC1.

For the rest of the KPIs (SL3, SL4, SL5) the results can be found under Section 4.5.2.a. These KPIs are not UC- or domain- specific and thus they do not need to be evaluated on the top of each UC. On the contrary, they evaluate the overall semantic layer (i.e., clarity of documentation, adaptability in different domains and number of entities/axioms provided).

b. **UC4 KPIs**

**KPI 1.** Comparing accuracy measure of face detection with accelerated AI model and standard AI model  
**- Description of demonstration:** Standard model and accelerated model will be trained on WIDER FACE train dataset, which is publicly open face detection benchmark dataset and is widely used in Computer vision research field. Then the performance of both models will be tested on WIDER FACE validation dataset in the experimental environment. The average percentage of activations of feature maps between the standard model and the compressed model will be compared using WIDER FACE validation set. The average precision of both models will be calculated and compared on WIDER FACE validation set. The method and other experimental environments to calculate average precision will comply with WIDER FACE benchmark.  
**- Results:**  
A parameter pruning and sparse connection method are implemented in order to accelerate the face detector. To induce the parameter pruning two objective functions are used in training phase and shown in equation 1. The L1 function is used to sparsify the feature map along with standard detection loss function. In equation 1 where the total number of layers is L, the size of the feature point map is C, the height is H, the width is W, and a pixel of the feature point map is represented by $x_{l,h,w,c}$, the first objective function applied the L1 objective function directly to each pixel of the feature map to sparsify it. As the second objective function, L1
objective function for channel pruning was used. After finding the maximum value for each channel from each layer, L1 objective function was applied and the lowest 20% of channels among all channels C were used in calculation. That is, it is assumed that a channel with low activation has little effect on detection performance, and a channel with high responsiveness has a high effect on the result. Therefore, the objective function is applied to induce sparsity only on the channels with low activation. Finally, α in front of the second function was used as a hyperparameter that weights between the two objective functions.

Equation 1. Sparsification loss.

\[
\sum_{l} \sum_{h} \sum_{w} \sum_{c} \left| x_{l,h,w,c} \right| + \alpha \sum_{i} \sum_{c \in C'} \max_{h,w}(x_{l,h,w,c})
\]

The neural network compression and acceleration training process was conducted in a two-stage pipeline. First, a standard neural network was trained with a WIDER FACE training dataset to find connectivity between parameters. In the second step, the compression and acceleration objective function was additionally applied to the objective function used in the existing neural network to remove some connections that do not significantly affect the performance of the neural network. For hyperparameters, α in Equation 2 was set to 0.1, and C’ was set to the lowest 20% channels with the lowest maximum value for training.

For validation set of WIDER FACE, performance was evaluated by calculating the percentage of activations and average precision (AP) as shown in Tables 1 and 2. The highest case for the percentage of activated feature map was 30.33%, where both objective functions were applied. When only channel pruning was applied, the highest case for activated channel was 20.00%, but the activated feature map was the highest. When both objective functions were applied, the detection performance was the worst as the feature point map was the highest.

<table>
<thead>
<tr>
<th></th>
<th>Size</th>
<th>The number of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>170.7 MB</td>
<td>42,748,434</td>
</tr>
<tr>
<td>Pruned Network</td>
<td>134.6 MB</td>
<td>33,680,043</td>
</tr>
</tbody>
</table>

Table 38. The size and the number of parameters of the original network and the pruned network.

<table>
<thead>
<tr>
<th></th>
<th>AP on easy set</th>
<th>AP on medium set</th>
<th>AP on hard set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.9431</td>
<td>0.9320</td>
<td>0.8569</td>
</tr>
<tr>
<td>Pruned network</td>
<td>0.9129</td>
<td>0.8966</td>
<td>0.7763</td>
</tr>
</tbody>
</table>

Table 39. Average precision on WIDER FACE validation dataset.

KPI 2. AI model substitution time

- **Description of demonstration:** We measured the AI model replacement performance of the DECENTER platform. For comparison, we measured the AI model replacement time when using the conventional method and the DECENTER method.
- **Results:** The following graph shows the comparison derived from the simulation.

![Comparison between original and DECENTER mode.](image)

We simulated changing the AI model by repeating 10 times using the existing method and the DECENTER method. On average, the DECENTER method took 15.94 seconds and the traditional method took 39.11 seconds. According to the experimental results, it was confirmed that, on average, a gain of 23.17 seconds was obtained when the DECENTER platform was used. Based on the results, it was found that it took only 41.22% of the time when using the DECENTER method compared to the existing method. Therefore, average reduction rate is 59.24%.

<table>
<thead>
<tr>
<th></th>
<th>Conventional Method(s)</th>
<th>DECENTER Method (s)</th>
<th>DECENTER/Conventional Ratio(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16.67</td>
<td>44.60</td>
<td>37.37</td>
</tr>
<tr>
<td>2</td>
<td>12.56</td>
<td>40.19</td>
<td>31.25</td>
</tr>
<tr>
<td>3</td>
<td>17.51</td>
<td>36.17</td>
<td>48.41</td>
</tr>
<tr>
<td>4</td>
<td>16.92</td>
<td>36.63</td>
<td>46.18</td>
</tr>
<tr>
<td>5</td>
<td>19.97</td>
<td>34.35</td>
<td>58.14</td>
</tr>
<tr>
<td>6</td>
<td>22.01</td>
<td>44.07</td>
<td>49.94</td>
</tr>
<tr>
<td>7</td>
<td>12.82</td>
<td>38.90</td>
<td>32.95</td>
</tr>
<tr>
<td>8</td>
<td>15.67</td>
<td>35.17</td>
<td>44.56</td>
</tr>
<tr>
<td>9</td>
<td>12.79</td>
<td>45.69</td>
<td>27.99</td>
</tr>
<tr>
<td>10</td>
<td>12.51</td>
<td>35.38</td>
<td>35.37</td>
</tr>
</tbody>
</table>
KPI 3. Accuracy verification of member verifier with AI model

**Description of demonstration:** Two common benchmark datasets are utilized as follows; 1) *YouTube Faces* which contains facial images of 1,595 different people with 63,800 training images and 7,975 test images, and 2) *FaceScrub* which also holds facial images of 530 celebrities with 67,177 training images and 2,650 test images. Each dataset configures two different groups and test images are employed to measure the member verification accuracy.

**Results:**
In order to validate the proposed method, we construct two different member verification AI models, which are trained with two face image datasets, respectively. Test images belong to the corresponding dataset are registered as member, while the others are classified as non-member.

![Member verifier example](image)

Any common Convolutional Neural Network (CNN) architecture such as AlexNet, VGG, ResNet are able to be utilized to extract face image descriptor, which holds discriminative face image features. In this case, we utilize VGG as baseline and additionally employ two more Fully-Connected (FC) layers as shown in the Figure 39. For the edge device, we fix the size of the image descriptor as 128-D, where the dimensionality of the first FC layer is 256, and the second FC layer is 128. The final output logits are obtained with Softmax operation to compute the probability of which group the input image falls into. When it comes to utilizing on the cloud, the dimensionality of each FC layer is increased by 2 times and also the number of FC layer is increased by 2 times, resulting higher capacity. As reported Table 41, the edge model only requires about 22% of the trainable parameters compared to the cloud model. Nevertheless, as shown in Table 42, the verification accuracy hardly decreases even if the model capacity is diminished in the edge.

<table>
<thead>
<tr>
<th># of params</th>
<th>Edge model</th>
<th>Cloud model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>65,792</td>
<td>295,168</td>
</tr>
</tbody>
</table>

*Table 41. The number of parameters (params) of the models of edge and cloud.*
### 4.5.3. Final Pilot Description

Ambient intelligence for office environments pilot was tested in KETI lab and all the devices set to be ready to work in a real environment. UC4 pilot test has proceeded with the DECENTER Platform.

This pilot test suggests a Smart Bulletin Board service on DECENTER platform. Instead of displaying the same information repeatedly, this smart bulletin detects an individual’s affiliation and displays only the appropriate information immediately. In this use case, the AI method in this use case is to detect an affiliation from the camera’s video stream.

![Camera](image)

*Figure 40. Camera used in UC4 to collect input data for the member verification process.*

When a person comes in front of the camera, content related to the meeting is displayed on the screen according to the member verification result.
When a person comes in front of the camera, content related to the meeting is displayed on the screen according to the member verification result. The pictures below are the screens the system can provide when the user stands in front of the camera (for KETI group, SNU group, and others).

**Figure 41. Example of person in front of the camera.**

**Figure 42. Example of content provided to visitors according to their membership.**

DECENTER provides methods to distribute AIs over cloud and edge with microservice architecture, which makes it easy to build an AI service over cloud and or edge. And in this pilot, we’re going to compose the service with three AI models. The First AI model will find the faces from the received video, the second AI model will extract features from detected faces for comparison, and the last AI model will do the comparison to find the affiliation of the person.
D5.2 Final release of the AI-integrated fog computing platform, demonstration KPIs and final setup of pilots for demonstrations

Figure 43. Example of App composer for UC4.

DECENTER provides a tool to build service architecture in an efficient way, the App Composer. With this GUI, we can request specific resources such as GPU or location, as well as other specifications to Kubernetes Deployment. We can also specify specific GPU resources to run the AI microservice and where to deploy the AI microservice.

Figure 44. Deployment of UC4 by App composer FogAtlas.

When the configuration of all the services is complete, and you can request to deploy them onto the cluster. The cluster used in this pilot consists of various nodes with different types of GPU resources that are placed in two regions.

Figure 45. List of microservices’ pods deployed on the DECENTER platform for UC4, showing their status, queried through kubectl Kubernetes command line interface.
This pilot is composed of three AI models - the one to detect faces from videos, another one to extract features from the detected faces, and the other one to find affiliation by comparison of those features, and we can check how they are working with this GUI. The incoming video and FPS of the first AI method are displayed on the far left, detected faces are displayed in the middle with the second AI, and the far-right displays the detected affiliation. Various metrics of serving an AI such as FPS, delay, or confidence score are displayed in this UI.

After checking the basic scenarios, additional pilots are conducted to change the model of specific microservices. If the 0.1 version model has been upgraded to 1.1, we can quickly change the model through the DECENTER platform. We tested and found that the model replacement time was reduced to less than 50% when using the DECENTER platform. This demo is used to validate KPI2 of UC4.
5. Conclusions

This deliverable collects the final outcomes of the activities of WP5, including the updates about the integration of the components from WP3 and WP4 and their application on the different use cases of the project. Furthermore, the final pilot scenarios and the demonstration of the use case KPIs have been updated and described in detail, showing clearly the benefits of the use of the DECENTER platform.

The document follows a similar structure of its initial version (D5.1) but including the last updates and modifications made from Y2 until the end of the project.

In section 2, the DECENTER technology is presented along with the final integration requirements of the DECENTER platform, including the platform services, the application services methods and the DECENTER tools.

Section 3 explains how to integrate the AI applications of DECENTER platform into the use-cases. Per each use case the details of the integration have been explained to show and help other interested people on applying the DECENTER platform to their own use case.

Finally, the section 4 includes the final description update of the use cases. Per each use case, the KPIs are described as well as the way to demonstrate each KPI. Then the results of each demonstration have been analysed and compared with the KPIs defined during the development of the project. In all cases, the objectives set have been met.

This document is written some weeks before the end of the project, but all the KPIs of the use cases still have been explained, demonstrated and their results have been analysed and justified. The use and the integration of the DECENTER platform through the four use cases have been explained focusing on the benefits of its use and comparing the results before the platform integration and after the platform integration. In conclusion, all KPIs have been successfully reached and every use case has found the benefits of the use of the DECENTER platform.

This deliverable reflects the joint effort from all European and Korean partners to collect and show the functionality and potential of the DECENTER platform at the final status of the project.
### 6. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6MR</td>
<td>Six-Monthly Report</td>
</tr>
<tr>
<td>CoA</td>
<td>Coordination Agreement</td>
</tr>
<tr>
<td>CA</td>
<td>Consortium Agreement</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CRD</td>
<td>Custom Resource Definition</td>
</tr>
<tr>
<td>DL</td>
<td>Deep Learning</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Networks</td>
</tr>
<tr>
<td>DoA</td>
<td>Description of Action</td>
</tr>
<tr>
<td>DPO</td>
<td>Data Protection Officer</td>
</tr>
<tr>
<td>EC</td>
<td>European Commission</td>
</tr>
<tr>
<td>ECRF</td>
<td>Effort and Cost Reporting Form</td>
</tr>
<tr>
<td>FS</td>
<td>Financial Statement</td>
</tr>
<tr>
<td>EU-GA</td>
<td>European Grant Agreement</td>
</tr>
<tr>
<td>GAS</td>
<td>General Assembly</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>IITP</td>
<td>Institute for Information &amp; communications Technology Promotion</td>
</tr>
<tr>
<td>IM</td>
<td>Impact Manager</td>
</tr>
<tr>
<td>IPR</td>
<td>Intellectual Property Right</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MSIT/IITP</td>
<td>Korean Ministry of Science and ICT</td>
</tr>
<tr>
<td>PM</td>
<td>Person Month</td>
</tr>
<tr>
<td>PMT</td>
<td>Project Management Team</td>
</tr>
<tr>
<td>PO</td>
<td>Project Officer</td>
</tr>
<tr>
<td>PR</td>
<td>Periodic Report</td>
</tr>
<tr>
<td>RP</td>
<td>Reporting Period</td>
</tr>
<tr>
<td>SoC</td>
<td>System on Chip</td>
</tr>
<tr>
<td>ToC</td>
<td>Table of Contents</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
</tbody>
</table>
7. Annex 1: KPI updates

Use case 1:

<table>
<thead>
<tr>
<th>Use Case 1 (Smart City)</th>
<th>KPI Name (description)</th>
<th>Qualitative/Quantitative</th>
<th>Metric</th>
<th>Requirements (Metric related statements)</th>
<th>Innovation (revised)</th>
<th>Method used to get the data</th>
<th>TS.3 Benefits brought by Decenter</th>
<th>TS.3 Evaluation of the KPI</th>
</tr>
</thead>
</table>
| N                      | Detecting key actors involved in the pedestrian crossing (for example: car, bus, truck, person, bicycle, motorcycle etc.) | Quantitative              | Latency [ms] | Smart crossing must support:  
  - low latency on detected actors to recognize eventually dangerous situations | With DECENTER it is possible to focus the latency problem using:  
  - AI Model deployed at the edge and Distributed AI Models  
  With DECENTER it is now possible to do that under than 100ms | Network latency and camera FPS can be measured to retrieve the latency in describing objects at the crossing. | Vertical resource orchestration plus AI model repository allow the system to react to different latency requirements | Changing the latency requirements: the microservices are orchestrated within the data-to-edge infrastructure. |
| 1                      | Ability to produce environmental data using IoT sensors to identify different environmental conditions | Quantitative              | Type of identified actors (i.e. number of different objects correctly identified) | More than 5 types of actors should be correctly identified (for example: car, bus, truck, person, bicycle, motorcycle etc.) UC requirements: FPS 6 | AI Model Repository: with DECENTER it is possible to update an AI model with a click  
Digital twin based on IoT data interpretation | The detected object are saved into a database, along with the type of each object. The log can be impacted to retrieve the identified objects. | AI model repository allow to implement and deploy different AI algorithms to detect and classify new objects | Updating the AI service loading a new model form the repository allows to detect new objects. |
| 2                      | Alert generation (performance of the system) | Quantitative / Qualitative | Response time [s] | The system has to trigger an alert in less than 100 milliseconds | With DECENTER it is possible to trigger actors:  
  - AI Model deployed at the edge and Distributed AI Models  
  - Multi-tier fog computing platform  
  - Virtual Resource orchestration to solve the problem of latency  
  - Improved security for the actors involved in the pedestrian crossing scenario (alerts triggered in less than 100ms) | The application response time is calculated as a difference between the timestamp of a received event and the timestamp of the actuation of the relative alerting device | Vertical resource orchestration allow to maintain the application QoS by moving services between cloud to edge | Changing the application latency requirements: the services are orchestrated in order to maintain a low application response time. |
| 3                      | Type of alert (red, yellow, green) | Quantitative | Type of alert (red, yellow, green) | The system must be able of classify the crossing conditions according to three danger risk levels (green low, yellow intermediate, red high) | With DECENTER it is possible to customize the performance of the edge AI model and the Cloud AI  
  - AI Model deployed at the edge and Distributed AI Models  
  - Multi-tier fog computing platform  
  - Digital Twin | 1) In the first phase we will inspect system log files to compare the performance of the edge AI model and the Cloud AI  
  2) Then, an AI will be exposed and it will be able to check the status of the system (red, yellow, green) and use it to check the results using the edge AI model or the Cloud AI | Resource orchestration, AI model repository and Digital Twin allow to change and monitor the behavior of the application | The alert level is continuously updated managing the data representing the Digital Twin.
Use case 2:

<table>
<thead>
<tr>
<th>N</th>
<th>KPI Name (description) (revised)</th>
<th>Qualitative/Quantitative</th>
<th>Metric</th>
<th>Requirements</th>
<th>Innovation (revised)</th>
<th>Method used to get the data (revised)</th>
<th>(T5.3) Benefits brought by Decenter</th>
<th>(T5.3) Evaluation of the KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Precision of the identification of the obstacle detected</td>
<td>Quantitative in two levels. First one: the output of the algorithm offers a percentage of confidence. Second one: the output can be a false positive.</td>
<td>Percentage of the precision identifying objects detected</td>
<td>In the last phase our goal is reach the 70% and then after 3 years 50%. <strong>UC Requirement</strong> FR_UC2_005, FR_UC2_006, NFR_UC2_001</td>
<td>Opportunity to apply different behaviors/answers to the situation of detecting and identifying an obstacle in the way using AI algorithms (AI Model deployed at the edge and Distributed AI Models)</td>
<td>Relation between the people/robots detected using the AI algorithm and the people/robots that are really present in the images</td>
<td>Decenter brings methods to use and switch easily AI algorithms and training sets</td>
<td>Reviewing the detected objects with the confidence level with the real object. Verify that the object recognition is not mistaken the classification</td>
</tr>
<tr>
<td>2</td>
<td>Accuracy of the estimation position of the obstacle detected</td>
<td>Quantitative. Difference between the real position of the object and the estimated position from the robot</td>
<td>Precision/Accuracy in cm.</td>
<td>The precision error of the position of the human/robots detected will not exceed 500mm in the test phase. Then after 3 years will not exceed 200mm. <strong>UC Requirement</strong> FR_UC2_005, NFR_UC2_009</td>
<td>Applying new methods of avoidance collision based on if the object is detected in front of the robot or is not on its path.</td>
<td>Relation between the calculated position of humans/robots detected using the scan sensor and the real position where they really are measured by hand</td>
<td>Decenter brings the object recognition and within the ability to fusion Depth data into the recognized zone and estimate the object distance</td>
<td>Comparing the detected measure with the real one</td>
</tr>
<tr>
<td>3</td>
<td>Saving CPU robot consuming</td>
<td>Quantitative. Difference the CPU robot consuming between using the baremetal architecture and using DECENTER infrastructure</td>
<td>Percentage of savings</td>
<td>The CPU usage of robots will be reduced by 10%. <strong>UC Requirement</strong> NFR_UC2_002, NFR_UC2_003, NFR_UC2_004</td>
<td>Using DECENTER offloading feature with the objective of reducing computational needs and costs</td>
<td>Monitoring the CPU usage in the robots.</td>
<td>Vertical and horizontal offloading capabilities of Decenter allows to reduce cpu usage. This leads to longer run time periods</td>
<td>Comparing the cpu saturation with offloading and without it</td>
</tr>
</tbody>
</table>
Use case 3:

<table>
<thead>
<tr>
<th>N</th>
<th>KPI Name (description) (revised)</th>
<th>Qualitative/Quantitative</th>
<th>Metric</th>
<th>Requirements</th>
<th>Innovation (revised)</th>
<th>Method used to get the data (revised)</th>
<th>(T5.3) Benefits brought by Decenter</th>
<th>(T5.3) Evaluation of the KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AI model response time performance</td>
<td>Quantitative</td>
<td>Response time [s]</td>
<td>Manager must be informed by the system for any security events within 1 second, for any less critical events within 5 minutes. UC Requirement: FR_UC3_008</td>
<td>Decenter’s Fog computing approach based on QoS-aware orchestration.</td>
<td>Estimates the time passed between a safety violation event (i.e. object detection/member verification) and the received notification. System logs with events’ timestamps will be inspected. QoS-aware orchestration of AI models</td>
<td>Measure the response time with and without QoS-aware orchestration.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Temporary use of external deployment options</td>
<td>Quantitative</td>
<td>Number of fog providers used in scenarios</td>
<td>Reduce operational costs UC Requirement: NFR_UC3_005, NFR_UC3_007</td>
<td>The open exchange of compiling resources in the Edge-to-Cloud continuum through Blockchain technology (i.e. smart contracts and oracles)</td>
<td>Economic estimation of the operational cost reduction in infrastructures: cost of ownership vs. cost of temporarily leasing</td>
<td>The Decenter Fog and Brokerage platform can offer users large quantity of infrastructures that can be leased by executing Smart Contracts</td>
<td>Economic estimation of the operational cost reduction in infrastructures: cost of ownership vs. cost of temporarily leasing</td>
</tr>
<tr>
<td>3</td>
<td>Improve Intelligence of smart constructions sites using on-demand AI models</td>
<td>Quantitative</td>
<td>Number of AI models employed</td>
<td>The smart construction environment must provide intelligence to the workers including the detection of vehicles, identification of persons, etc. UC Requirement: FR_UC3_001 to 7, NFR_UC3_001</td>
<td>Scalability of available on-demand AI models achieved by using Decenter’s orchestrator and AI Model repository</td>
<td>Counting the used AI models from the AI model repository</td>
<td>Repository of pretrained AI Models</td>
<td>Number of AI models that can be implemented in different construction site scenarios</td>
</tr>
<tr>
<td>4</td>
<td>Improve privacy, safety and security in smart construction sites</td>
<td>Qualitative</td>
<td></td>
<td>The actual persons at the construction site must be detected and identified UC Requirement: FR_UC3_001 to 7, NFR_UC3_005, NFR_UC3_007</td>
<td>Smart contracts and oracles used to facilitate secure access to AI models and achieve transparency and traceability when using the AI models</td>
<td>Leveraging blockchain technology for transparent and secure operations over AI model repository</td>
<td>Specially designed Smart Contracts and Smart Oracles</td>
<td>Number of biometric models, number of smart contract transactions</td>
</tr>
</tbody>
</table>
### Use case 4:

<table>
<thead>
<tr>
<th>N</th>
<th>KPI Name (description) (revised)</th>
<th>Qualitative/Quantitative</th>
<th>Requirement Metric</th>
<th>Metric related Requirement</th>
<th>Innovation (revised)</th>
<th>Method used to get the data (revised)</th>
<th>(T5.3) Benefits brought by Decenter</th>
<th>(T5.3) Evaluation of the KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Comparing accuracy measure of face detection with accelerated AI model and standard AI model</td>
<td>Quantitative</td>
<td>Average Precision / size of parameters of AI model</td>
<td>The average precision of accelerated AI model should be over 80% of the original ones in the WIDER FACE benchmark dataset. The number of parameters and the size of the accelerated AI model should be reduced down at least 80% of the original ones.</td>
<td>Decenter can provide the accelerated AI model which has smaller model size while keeping the AI performance</td>
<td>Image data is achieved through camera in the use case.</td>
<td>Using the AI optimization technique from T4.3, AI model can be accelerated while keeping the detection accuracy.</td>
<td>Evaluation with benchmark public Test DataSet (Calculate the average precision of standard AI model and accelerated AI model evaluated on WIDER FACE benchmark publicly open dataset and calculated the size of parameter of both AI models.)</td>
</tr>
<tr>
<td>2</td>
<td>AI model substitution time</td>
<td>Quantitative</td>
<td>Time ratio (%) Time with DECENTER method/ Time with previous method</td>
<td>AI model substitution time with DECENTER platform will be better than the previous method. We measure the time to substitute AI model with DECENTER.</td>
<td>The DECENTER platform provides efficient methods to substitute AI models in running microservices. When an AI model update is required, the microservices at the edge fetch the appropriate AI model from the AI model repository.</td>
<td>Compare average time for AI model substitution</td>
<td>The DECENTER platform provides efficient methods to substitute AI models in running microservices. When an AI model update is required, the microservices at the edge fetch the appropriate AI model from the AI model repository.</td>
<td>Evaluation with Demonstration (Count the number of AI service on the same service)</td>
</tr>
<tr>
<td>3</td>
<td>Accuracy verification of member verifier with AI model</td>
<td>Quantitative</td>
<td>Accuracy</td>
<td>The accuracy of the member verification should be over 80% in the Labeled Faces in the Wild (LFW) benchmark dataset.</td>
<td>DECENTER can provide a stable member verification service without additional registration of personal information.</td>
<td>Image data is achieved through camera in the use case.</td>
<td>It can successfully verify the member while preserving the privacy.</td>
<td>Evaluation with benchmark public Test DataSet (Calculate the accuracy of AI model evaluated on the Labeled Faces in the Wild (LFW) benchmark publicly open dataset)</td>
</tr>
</tbody>
</table>