EXTRACT

A distributed data-mining software platform for extreme data across the compute continuum

D1.3 Final release of the EXTRACT use-cases

Version 1

Documentation Information

Contract Number	**101093110**
Project Website	www.extract-project.eu
Contractual Deadline	30.06.2025
Dissemination Level	PU
Nature	OTHER
Author	OBSPARIS
Contributors	LRI, BSC, MATHEMA, IBM
Reviewer	BSC
Keywords	Semantic Urban Digital Twin, Astrophysics, Extreme Data, Emergency Preparedness, Personalized Evacuation Route

The EXTRACT Project has received funding from the European Union's Horizon Europe programme under grant agreement number 101093110.

EXTRACT
A distributed data-mining software platform for extreme data across the compute continuum

Change Log

Version	Description Change
V0.1	ToC - First Draft
V0.2	Executive Summary - Second Draft
V0.3	Detailed Breakdown of updates for the PER Use-Case
V0.4	Detailed Breakdown of updates TASKA Use-Case
V0.5	TASKA Overall Revision
V0.6	PER Overall Revision
V0.7	PER use-case & TASKA use-case – Next steps
V0.8	Conclusions
V0.9	Review by BSC
V1.0	Submitted Version

1. Executive Summary

This deliverable presents the final release of the EXTRACT use cases and the technologies developed for them in Phase 3, concluding the development activities initiated in Deliverables D1.1 and D1.2. It provides a comprehensive account of the implementation, integration, and validation of the two pilot cases designed to demonstrate the EXTRACT platform's capabilities in extreme data mining across the compute continuum:

- **PER** (Personalised Evacuation Route), addressing citizen-centric emergency management in complex urban environments.
- **TASKA** (Transient Astrophysics with a SKA Pathfinder), focusing on real-time scientific data analysis in the field of radio astronomy.

The deliverable also features technical models and deployment architectures, and includes links to the relevant software GitLab repositories, supporting reusability, open science, and collaborative development.

The EXTRACT platform responds to the growing need for scalable and trustworthy data mining architectures capable of operating across heterogeneous, secure, and energy-efficient infrastructures spanning from the edge to high-performance computing (HPC). Both use-cases have now reached an advanced stage of implementation, reflecting the platform's capacity to support demanding, real-world applications.

In its final release, the **TASKA** use-case delivers a modular, Jupyter-based framework for real-time data processing in astrophysics. Building on the Minimum Viable Product (MVP) introduced in D1.2, it now includes multiple validated workflows, each aligned with a distinct sub-use-case (A1, A2, C, and D). These workflows collectively demonstrate capabilities in handling real-time observations, event trigger detection, semi-supervised data reduction pipelines, and deep learning for radio image reconstruction using data from the NenuFAR radio telescope. All software components are developed according to clean code principles, support Docker-based containerisation, and are designed to ensure long-term sustainability and scalability.

The final release of the **PER** use-case has achieved full integration of a semantic Urban Digital Twin (UDT), a Reinforcement Learning (RL) engine, and a simulation environment for modelling pedestrian movement, thus enabling adaptive and personalised evacuation strategies in dynamic emergency scenarios. Key technical advances include the deployment of a microservices-based architecture and a 5G Multi-access Edge Computing (MEC) prototype, supporting low-latency, secure communication across edge and cloud layers. The introduction of the Evacuation Mobile App (EMA) enables direct user interaction and real-time route guidance, further enhancing the system's practical applicability in operational contexts.

Both use-cases confirm the feasibility, flexibility, and scalability of the EXTRACT platform for generating and deploying actionable insights across distributed computing infrastructures, while meeting stringent requirements related to latency, security, and energy efficiency. This deliverable complements the platform developments carried out in WP2, the infrastructure adaptations in WP3, and the orchestration strategies elaborated in WP4, and it contributes to the preparation of final piloting activities and long-term sustainability planning in the closing stages of the project.

To facilitate clarity and ensure self-sufficiency, this document is designed as a standalone deliverable. Where appropriate, selected content from previous deliverables—particularly D1.2—has been retained or adapted. This approach enables a comprehensive and coherent understanding of the final implementation of the use-cases without requiring reference to earlier documents.

1.1. Relation with other Deliverables and Tasks

This deliverable concludes the activities conducted in WP1 and finalizes the development and validation of the EXTRACT use-cases—PER and TASKA. It reflects the culmination of the iterative development process that began with requirements elicitation in D1.1 and progressed through MVP implementation and early validation in D1.2. As detailed in D1.2, the initial MVP phase focused on deploying MVPs for both PER and TASKA. For TASKA, this involved establishing a workflow architecture for radio astronomy data imaging, enabling efficient processing and data management by treating data as 'objects'. For PER, the MVP integrated a semantic Urban Digital Twin, a Multi-Agent Reinforcement Learning model, and a simulation environment to develop adaptive evacuation strategies for complex urban settings like Venice, validated through initial field tests. This phase also outlined core functionalities and technical strategies as a foundation for ongoing development.

The final release of the use-cases builds upon and contributes to the integrated EXTRACT platform by:

- Demonstrating the deployment **of unified data mining workflows** addressing the specific extreme data characteristics of the two use-cases;
- Validating system performance across the **compute continuum**, from edge to cloud and HPC resources, and aligning with key use-case KPIs;
- Showcasing how the platform supports **secure, energy-efficient, and real-time data processing**, tailored to the operational needs of urban crisis management and astrophysics research.

This deliverable is closely interlinked with the technical outcomes of the project and supports the objectives and evaluations reported in:

Deliverable	Tasks	Relation
D2.3	T2.3	D3.3 utilizes the finalized data-mining framework described in D2.3 and provides orchestration and monitoring capabilities that ensure effective deployment, execution, and monitoring of workflows defined using the EXTRACT data-mining framework. In that deliverable, we also describe how COMPSs integrates Lithops jobs together.
D3.2	T3.2, T3.3	D3.3 is the final version and enhancement of D3.2, fully developing and integrating the orchestration mechanisms (T3.2) and the distributed monitoring infrastructure (T3.3) initially presented in D3.2.
D4.3	T4.2, T4.3	This deliverable leverages the final version of the compute continuum abstraction layer (D4.3), integrating the interoperability

		mechanisms (T4.2) and cybersecurity functionalities (T4.3) to provide robust and secure orchestration and monitoring across diverse computing environments.

In this final stage, particular attention has been paid to interoperability, system maturity, and readiness for downstream exploitation (WP5), with the use-cases serving both as **validation environments** and as **demonstrators of real-world impact**.

1.2. Structure of the Document

This document is structured to provide a comprehensive overview of the final release of the EXTRACT use-cases and their components.

- The **Executive Summary** offers a high-level synthesis of the deliverable, outlining its main contributions and connections with related technical work packages.
- **Section 2** provides an updated framing of the use-cases in the context of **extreme data scenarios**, highlighting their relevance as testbeds for the EXTRACT platform.
- **Section 3** presents the **background and motivation** for each use-case, including the evolution from MVPs to mature implementations, and updates the detailed scenarios for TASKA and PER.
- **Section 4** describes the **technological framework** supporting the use-cases. It covers data acquisition, processing, orchestration, and analytics components, emphasizing the innovations and technical complexity embedded in the final deployments.
- **Section 5** outlines **next steps and potential enhancements**, including ongoing validation, stakeholder engagement, and alignment with sustainability and exploitation pathways.
- **Section 6** concludes the document by summarizing key outcomes and reflecting on the contributions of the use-cases to the project's overall objectives.
- **Section 7** provides a list of **acronyms and abbreviations**.

2. EXTRACT Project Overview

2.1. Project Scope

In the digital era, data has become the cornerstone of transformative processes across diverse sectors. However, conventional data mining strategies, although adept at handling specific data needs, often struggle when the data attributes venture into extreme territories. The emergence of extreme data characteristics presents a formidable challenge that necessitates innovative, comprehensive solutions. These solutions need to foster the design, deployment, and streamlined execution of data mining workflows across a diverse, secure, and energy-efficient computing landscape, while addressing the unique demands of extreme data.

The EXTRACT project sets out to tackle this technological void by introducing a data-driven, open-source software platform. This platform amalgamates state-of-the-art technologies, promoting the development of trustworthy, accurate, fair, and eco-friendly data mining workflows capable of generating actionable and high-quality insights. Targeting the entire lifecycle of extreme data mining workflows, the EXTRACT platform prioritizes enhancements in performance, energy efficiency, scalability, and security while while addressing the challenges posed by volume, velocity, and complexity of extreme data

2.2. Extreme Data in PER and TASKA Use-Cases

The platform's effectiveness and capabilities will be examined through two real-world scenarios, each with distinct extreme data requirements:

The Personalized Evacuation Route (PER) - This use-case unifies data from European services like Copernicus and Galileo, 5G localization signals, and IoT sensors embedded in smart city infrastructures, focusing on civilian-centric crisis management.

Disasters in this scenario trigger sudden and large data spikes. For example, a scenario such as a large fire in an urban context and the consequent customized collective evacuation mechanism generates a data flow that can be considered extreme in terms of volume, heterogeneity and dynamics.

The fundamental challenge lies in the real-time processing and analysis of this data surge to produce effective evacuation strategies.

Transient Astrophysics using the SKA pathfinder (TASKA) - This scenario processes extreme volumes of data from radio-telescopes (especially NenuFAR). The EXTRACT prototype is designed for real-time solar activity assessment, enabling further scientific exploration as well as proposing a new framework for heavy data processing.

This use-case constantly handles vast amounts of data, making it "extreme" due to its substantial size and continuous need for real-time interpretation. The primary challenge with TASKA is to manage, process, and extract meaningful insights from these uninterrupted, colossal data streams.

To meet these challenges, the EXTRACT platform integrates a spectrum of computing technologies, from edge to cloud to high-performance computing (HPC), forming a unified, secure computational continuum. It incorporates enhanced data infrastructures, AI and big data frameworks, novel data-driven orchestration, and distributed monitoring mechanisms, providing a unified continuum abstraction and ensuring cybersecurity and digital privacy across all its layers. The main objective is to provide a virtually local working environment for researchers, hiding all the logistical burden away from the eyes of the user.

3. Background and Motivation

3.1. TASKA Use-Case Scenario

3.1.1. General Background and Requirements

Modern astronomy observatory (Like LOFAR, NenuFAR and the future SKA) working in the radio band are producing a tremendous deluge of raw data (consisting of billions of measurement points) collected from observation of the Universe. To achieve the objectives of the scientific program and obtained reliable scientific products, data processing and product quality/fidelity assessment are necessary before delivering the scientific products to the researchers. To minimize the burden both manually and through heterogeneous computing platforms of data reduction and through heterogeneous computing platforms, the scientific community requires the development of a dedicated framework for decision-making and workflow composition that can perform time-consuming task efficiently in a record time. Modern data recording and processing require a level of decision-making to set up the instrument in the correct mode (to reduce the data volume as well as the impact on resources), or decide on the data reduction strategy to follow, especially if the target of interest is displaying various levels of variability (e.g., Solar activity). In the scope of the EXTRACT project, TASKA use-cases aim at addressing the requirements of the observation of such sources, by focusing on the "Space Weather" case which monitors the solar activity in radio.

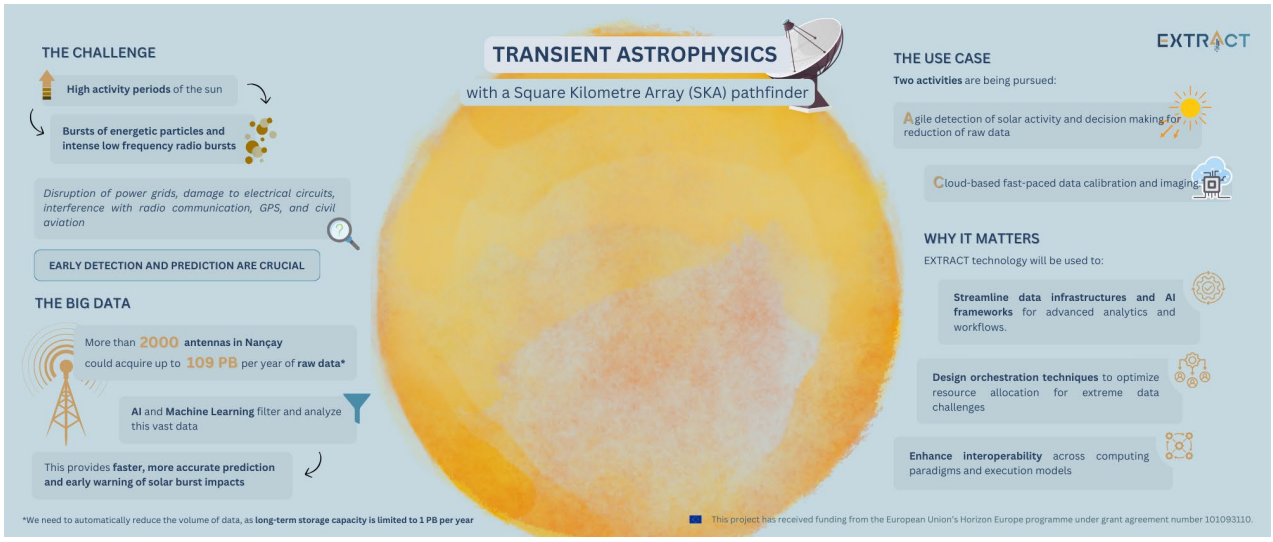

Figure 1: TASKA use-case – Transient Astrophysics with an SKA Pathfinder. *This infographic illustrates the scientific and technological challenges addressed by the TASKA use-case, including the detection of solar radio bursts and the management of extreme data volumes produced by next-generation radio telescopes. It highlights how EXTRACT technologies enable early prediction and fast data processing through AI-based filtering, modular workflows, and cloud-based orchestration. The approach supports timely scientific insight and improves the resilience of systems affected by space weather phenomena.*

Astronomy has a set of specific requirements that must be met throughout the development process:

- **TASKA use-case expected overall performances**

For the entirety of the EXTRACT project, the main objectives are: i) to devise high-performance data reduction workflows using of-the-shelf technologies or dedicated developments, ii) implement new features in instrumental configuration automation, iii) enable manipulating scientific data in higher level of abstraction than regular scientific data management to handle the extremely distributed data

- **Data abstraction & user interaction/user agency**

The multiplicity of data files produced by radio receivers as well as their volume on disk, slow down the efficiency to produce scientific results, especially if manual tasks have to be carried out by scientists or data processing non-experts. The objective is to take away the burden of the data by considering data elements as "objects" rather than a pile of files. The scientist should be able to handle a dataset, regardless of its complexity and be able to apply a series of processing steps **seamlessly** to **quickly access intermediary or end products**. The user must also be able to implement "logic" inside the workflow based on semi-interactive decision-making (e.g. "do we need to perform this task?", "shall we stop looping?") where the user has to interact remotely with the suspended workflow with "Stop", "continue" signals. The user interface is also key on top of the underlying machinery that handle the data and the workflow.

- **Data integrity**

The specifics of astronomy in general, and radio astronomy in particular, is to guarantee the accuracy and the precision of the final measurements, contrary to video restoration/denoising which case about the abstract rather than a publishable together. Deliberate alteration or distortion of the data should be done with specific knowledge as part of data processing steps (e.g.cleaning, data averaging, calibration or transformation). No side process should impact artificially the quality of the measurement nor the reproducibility of a scientific product in an uncontrolled way. Compared to other data restoration problems (e.g., video restoration), TASKA end-users value more the scientific value and quantitative correctness of the products than the qualitative performance that can be brought by the methods (resolution, data continuity): e.g., scientists would rather have a correct (but degraded) image rather than a sharper (but biased) image.

This principle conducted the technological developments carried out during the TASKA project.

- **Data Security**

The tools that emerge from TASKA will be used in the context of a multi-purpose instrumental environment. Hopefully, many scientific teams will be required to use common tools. Scientific privacy should be ensured between scientific teams to enable exclusivity of the data (observed data are proprietary for a period). Compared to PER use-case, no personal data is involved during the data acquisition. However, sensitive information about the targets or the data processing recipes should stay within the perimeter of a given team until publicly released eventually through publication (e.g. embargo on sources)

- **Science reproducibility**

Scientific production is lead by the principle of reproducibility. Long-term studies and the fast-evolving development of new data processing methods push the scientific community to enable the capability to regenerate results, as part of stability but also peer investigation. The dataset that undergo data reduction will produce scientific products that can last for years as reference. They can also be compared to similar products created with other recipes, other workflows, other tools. As a matter of being able to preserve the scientific value of a data reduction or of a product, it is mandatory that all the information that resulted in a given products should be recreated independently and without ambiguity by a third party. For this purpose, a data catalog embeds both the location of the data files as well as metadata that tracks the relational links between files and the lifecycle of the data through the workflow (referred to as "provenance").

3.1.2. use-case A: Real-time detection for radio beamformed observations

The goal was to develop a simple AI/ML classifier to the current data flow of NenuFAR (Beamforming mode) in order to detect and classify "events" and adjust the receiver parameter to an optimal resolution before the recording of the data on disk. Thanks to an independent development carried at Observatoire de Paris, we adapted MurMure (Modulate Multicase Receiver) which enabled the implementation of such feature.

In the final release, use-case A was split into use-case A1 and A2.

- **use-case A1:** A dedicated and automated detection method to detect and classify Solar radio spikes, using a deep-learning based convolutional neural network named SpikeNet.
- **use-case A2:** A dedicated burst detector trained for Jupiter millisecond bursts (S-bursts), it takes advantage of a detection method based on Fast Fourier and Radon transforms.

An adaptation of MurMure was developed close to the NenuFAR receiver (UnDySPuTed) to detect and classify events (from the Sun and Jupiter) on the full-resolution data stream coming from the telescope ("LaNewBa"), so as to enable the trigger of various data record resolutions set upon science content. NenuFAR/UnDySPuTed is currently built using the GPU technology thus appropriate programming models will have to be used for this development on the edge component. The final goal is to process data on the fly based on classification (clean and integrate) including short / medium / large scale events (including time-frequency feature polygons), radio interferences (RFI), quiet (background) regions, etc. The tool was developed through an off-line pipeline (better for tracking pipeline parameters and reproducibility), with the goal to enable classification storage (analogue-to-information) feeding an RFI/Science information database. This database is already in itself a valuable scientific product. Based on the development and knowledge gathered from the Event detection system, the front-end system (LaNewBa) will also be improved in future developments with an automated faulty antenna stream detector, in order to detect and blank faulty signals before they are merged into the raw data stream fed into UnDySPuTed.

As of June 2025, A1 is successfully running on real-time observations of the Sun, leading to a reduction of a factor 10 in high-resolution data. A2 is completely migrated from IDL to Python and adapted within MurMuRe, some

processing steps need to be adapted for NenuFAR before real-time application. The final release incorporates A1 and the remaining of the project will focus on the development of A2, easily adaptable to be integrated along with A1.

3.1.3. use-case C: Workflow orchestration for radio interferometry

Most of the work in use-case C has been covered during the period between D1.1 and D1.2. The description covered in D1.2 has been implemented following the MVP definition. The overall architecture of the edge/cloud system which covers use-case C has been implemented but is in constant improvement. The main motivation for use-case C is to provide provide end-users with the capability to process radio interferometric data (from NenuFAR but not only) on a seamless platform that enables: resource management and optimization, control and interactivity with users, abstract storage of data and metadata ingestion, simplicity of implementation, processing time optimization. The performance of the platform (particularly beyond the MVP) is critical for an end-to-end processing of astronomical data in the scope of the conditions defined in 3.1.1. This period use-case development are central to move from the implemented MVP to real-life on various scientific applications using similar data structure. For this, the **Undulatus** framework has been developed and is further presented in the next section. Additional features were added: i) capability to suspend the workflow while running and ask the operator for a decision (i.e. STOP/CONTINUE) before pursuing, ii) the capability to run computationally-demanding tasks in a HPC environment, iii) hosting on multiple site behind object storage.

The final release contains all minimal examples that illustrates the features of use-case C. The final period of the TASKA use-case C project will be devoted to the "fine tuning" and to the optimization of resources from the current architecture as well as the final definition of the data catalog (which must cover all possible applications)

3.1.4. use-case D: Time-agile imager for radio interferometry

In the scope of use-case C, now ready for production on the optimization of existing imaging pipelines, this use-case proposes to go a step further, with innovative image reconstruction for dynamical sources. Classical interferometric imaging is based on a frame-by-frame reconstruction, with careful calibration and artifact processing. However, in case of intense and fast-varying radio sources, such as the Sun, such algorithms proved to be rather inefficient, mostly because the dynamical radio source overcomes the calibrators. We released a prototype new imager, **Cassey**, based on previous developments focusing on machine learning video-reconstruction. In such a framework, the series of frames are reconstructed at once, tracking the moving structures throughout the frames. Ultimately, this new algorithm should operate directly on interferometric visibilities (i.e., on the spatial Fourier domain), in order to reduce artifacts introduced by the costly gridding/degridding step of classical imaging pipelines.

As of June 2025, the Cassey algorithm & network exists and the last 6-months period will be devoted to the training and fine tuning of the network on realistic or real data. developed on the cloud part for the training and will use the model serving (further described in Deliverable D4.2) when applied to the data.

3.1.5. use-case E (optional)

Use-case E is closing the loop of all TASKA use-cases: use-case A is about event detection (in the temporal-spectral domain) and data flow optimization; while use-cases C and D are dealing with imaging, with a specific focus on ML processing in use-case D. Using the model-serving tools, use-case E proposes to implement feature recognition and dynamic imaging in Nançay (edge), and add a feature matching an algorithm that merges the temporal-spectral features (use-case A) and the dynamical/moving radio sources (use-case D), allowing us to add spatial tagging to the use-case A outputs.The resulting knowledge base (with temporal-spatial-spectral features) would be a huge step forward in the understanding of Solar radio sources in NenuFAR's spectral domain.

As of June 2025, this optional use-case is placed as a "nice to have" feature and the last 6 months of the project will be the time to properly set the specifications for this mode. The latter depend on the actual performance and limiting factors of the use-case A1, A2, C and D.

3.1.6. TASKA Final release

The TASKA MVP was mainly focused on the first implementation of TASKA Use-case C, proposing a workflow architecture for radio astronomy interferometric data imaging. In the final product, all development bricks and software exist in a pre-release state (A1, A2, C and D, except E), but still require fine-tuning. The final 6-months period is now very relevant to increase the robustness of the products, apply it to real situations and optimize their parameters with respect to the real situations.

3.2. PER Use-Case Scenario

3.2.1. General Background and Requirements

The PER pilot focuses on the historical city of Venice, a city marked by exceptional complexity in its urban fabric, including high tourist density, fragile infrastructure, and limited evacuation routes constrained by narrow alleys, bridges, and waterways. These conditions pose significant challenges for traditional evacuation strategies, which typically rely on static, generic instructions. PER instead introduces a model of **personalized evacuation**, where each individual receives dynamic route suggestions based on their location, mobility, surrounding crowd density, and environmental factors. This approach improves safety, reduces bottlenecks, and enhances resilience in emergency response scenarios.

Figure 2: *Infographic: Personalized Evacuation Route – A Smart Solution for Crisis Management in Venice. This infographic illustrates the PER use-case, which leverages the EXTRACT platform to generate personalized evacuation routes in real time during urban emergencies. It highlights Venice's complex geography and high flooding risk, and shows how the integration of Urban Digital Twins, IoT data, and reinforcement learning enables adaptive, citizen-centric crisis response through mobile guidance.*

Requirements

The functional, performance, and integration requirements guiding the development of the PER MVP were defined in *Deliverable D1.2* and remain fully valid in this final release. These requirements have shaped the architecture and implementation strategy, ensuring the system's capacity to support real-time, personalized evacuation planning in complex urban environments. While no new requirement categories have been introduced, the refined architecture presented in this deliverable—particularly the integration of the **5G MEC component** and the **Evacuation Mobile App (EMA)**—adds further considerations related to edge deployment, real-time communication, and user interaction. **These enhancements are detailed in Sections 4.2 and 4.2.3.**

The original requirement breakdown is recalled below for completeness:

Basic Functional Requirements

These involve verifying the feasibility of the core features characterizing the PER application:

- **Urban Data Integration and Hazard Detection**: The MVP is designed to collect simulated data representing inputs from the urban environment and integrate them using a semantic approach.
- **Personalized Evacuation Routes**: Leveraging simulated data from Venice's unique urban landscape, the system generates personalized evacuation routes using RL, accounting for both individual and environmental variables.
- **Simulator**: Simulation of individual behavior and emulation of mobile device data (e.g. GPS coordinates) for people moving within a selected area of the city.

Performance Requirements

These focus on verifying the scalability of the technological approach adopted:

- **Simulated Georeferenced Individuals**: Simulating a large number of individual positions across Venice is essential for (a) early technology performance evaluation, and (b) dataflow testing in the MVP. It is also critical during the RL training phase.
- **System Performance**: The MVP is implemented with performance considerations to meet the experimental demands of testing and training.

Integration Requirements

These focus on verifying that the designed dataflow among components operates as intended:

- **Integration APIs**: Seamless integration with the UDT, RL engine, and Simulator is central to the MVP's functionality.
- **Data and Machine Learning Support**: The system relies on libraries capable of processing large datasets and supporting advanced machine learning workflows for informed, timely decision-making during emergency events.

Datasets

The datasets used in the PER MVP remain consistent with those described in *Deliverable D1.2, Section 4.2.1*. These include both **static datasets** (e.g. building layouts, urban infrastructure) and **dynamic datasets** (e.g. crowd flows, environmental conditions), which are integrated via the UDT to support real-time simulation and personalized routing.

> ❖ **In the current release, data flows have been fully implemented across the platform components—including ingestion, cataloging, processing, and notification—through the EXTRACT Data Layer and the DMF, as detailed in *Deliverables D2.2* and *D2.3*.**

While no new datasets were introduced in this phase, **semantic alignment and orchestration of data usage** have been refined to support system-level interoperability and readiness for pilot execution.

More details, including a full description of the use case rationale, scenario design, and MVP implementation, are provided in Deliverable D1.2.

3.2.2 PER use-case Final Architecture and Loop Integration

The architecture of the PER use-case builds on the MVP design described in *Deliverable D1.2, Section 4.2.2*. That version introduced the core components of the decision-support system: the UDT, the People Movement Simulator (SIM), the Multi-Agent Reinforcement Learning engine (RL), and the Evacuation Mobile App (EMA).

Since then, the focus has shifted from component-level development to **full integration and operational loop closure**. The four components now function as a coherent pipeline, where inputs from the UDT and SIM dynamically inform RL-based decision-making, and updated evacuation routes are delivered in real time via the EMA.

Key advancements in this phase include**:**

- Semantic alignment of data flows between UDT, SIM, and RL;
- Integration of the simulation environment to support both RL training and testing;
- Implementation of the EMA interface with adaptive routing logic and real-time updates;
- Confirmation of feedback loop closure, allowing continuous system refinement based on simulated or live user behavior.

This end-to-end integration ensures that evacuation decisions can evolve in real time, based on changing conditions and user profiles. The system has reached a maturity level suitable for piloting and stakeholder demonstration, with potential for adaptation to other urban environments beyond Venice.

3.2.3 Outlook and Transferability

The PER use-case is poised for validation with a live test planned in Venice during the first week of October 2025. This exercise will involve trained volunteers—including local citizens and retired police officers—and will evaluate the system's performance under simulated emergency conditions, contributing to EXTRACT's broader piloting and stakeholder engagement strategy. A cohort of volunteers has already been recruited and trained on the Evacuation Mobile App (EMA) , aiming to provide critical insights into both system usability and the effectiveness of personalized evacuation strategies in real-time scenarios.

Figure 3: *Group photo of the volunteers from the city of Venice participating in the EXTRACT PER use-case.*

Although full-scale testing is pending, the PER architecture, with its integrated Urban Digital Twin (UDT), People Movement Simulator, Reinforcement Learning (RL) engine, and EMA, is fully functional. This establishes a robust foundation for realistic evaluation and further iteration. The PER use-case demonstrates the potential of the EXTRACT platform to enable data-driven, personalized emergency response in complex urban settings. By combining heterogeneous data sources, semantic modeling, adaptive routing, and mobile communication, the system promotes both citizen safety and urban resilience.

> ❖ **The PER use-case demonstrates the potential of the EXTRACT platform to enable data-driven, personalized emergency response in complex urban settings. By combining heterogeneous data sources, semantic modeling, adaptive routing, and mobile communication, the system promotes both citizen safety and urban resilience.**

With full integration completed and real-world testing imminent, the PER use-case is now ready for validation in operational conditions. Its modular design and semantic architecture ensure transferability to other urban contexts—especially those exposed to climate-related risks, high tourist density, or constrained infrastructure. The results of the Venice pilot will inform further technical refinements and support broader replication, contributing to EXTRACT's sustainability and exploitation roadmap.

4. Final Architecture Integration

This section presents the final integration of the EXTRACT platform across the two use-cases—**TASKA** and **PER**—highlighting the implemented architecture, technologies used, and functional readiness for real-world piloting. It outlines how the individual components developed in earlier phases have been brought together into cohesive, end-to-end workflows capable of handling the challenges posed by **extreme data environments**.

While the MVPs served as a critical foundation for early validation (as described in D1.2), the current focus is on the **full-scale implementation** of the EXTRACT architecture. This includes demonstrating interoperability, data flow consistency, orchestration logic, and computational performance across the data lifecycle—from ingestion to analytics and decision-making.

The sections that follow detail the final state of system integration for each use-case, emphasizing both common architectural elements and domain-specific adaptations.

4.1. TASKA Final release

4.1.1. Use-case A final specifications

New development occured since the release of the MVP (which only focused on use-case C). Real-time detection of radio spikes (from the Sun or Jupiter) and receiver dynamic parametrization is only possible thanks to an original development in

Observatoire de Paris of the *Modulate Multicast Receiver* (MurMuRe, check: https://gitlab.bsc.es/extract/extract-use-cases/taska/use-case-a/modular_multicast_receiver). This module takes advantage of th CuPy library and allows to navigate easily between CPU and GPU memory for computing data as it is being acquired by the telescope. See Figures 4.1.c to e.

- **use-case A1:**

The SpikeNet network, described in Figure 4.1.a, has been implemented and adapted to be inserted in the NenuFAR workflow and back-ends.

We successfully managed to apply the network to real-time solar observations taken with the NenuFAR radiotelescope resulting in a successful 10x volume compression by keeping the full resolution when spikes are present and accumulating and averaging at lower record rate when no detection is declared. In Figure 4.1.f, we show the results of these observation, the tiles where spikes were detected are overplotted in yellow. These tiles are saved at high resolution, while the whole observation is compressed at lower resolution, still allowing the study of slowly varying emissions.

Output files generated by the network are set of tiles containing detections and physical properties of each spike, all gathered as HDF5 files.

The code is available at : https://gitlab.bsc.es/extract/extract-use-cases/taska/use-case-a/A1_SpikeNet

- **use-case A2:**

What can be done for short solar spikes could also be applied to other planetary transients such as Jupiter. However, the dynamic spectral features presents various shapes depending on the in-situ physical status of the source. Based from a method developed during a PhD work (E. Mauduit) on an another radio telescope, the Nançay Decameter Array (NDA).

The original IDL algorithm has been converted to Python to enable code integration in the data flow.

The code is public and available at : https://gitlab.obspm.fr/extract/taska-a2.

A first application of Jovian radio emissions through this system with NenuFAR is currently under validation. Operation such as identifying the Jovian bursts as well as discarding Radio Frequency Interferences (RFI) which interfere with the Jovian bursts are being adapted to NenuFAR, see Figure 4.1.b.

This algorithm is being validated and applied to on-disk NenuFAR observations of Jupiter in order to build a labelled database that will be use to train a dedicated neural network. This network will be based on anomaly detection since a classification algorithm is not adapted due the large diversity of shapes in S-bursts.

After validation, the algorithm will be integrated in the same framework than A1 and will be subject to fine tuning before being released as a dedicated neural network for detecting Jovian bursts.

Output products will also contain the location of the bursts as well as the physical parameters of the bursts (drift rates, lengths, frequency localization) in the form of HDF5 files.

For both use-cases, we will look into more universal, and efficient, formats for stocking such informations on disk.

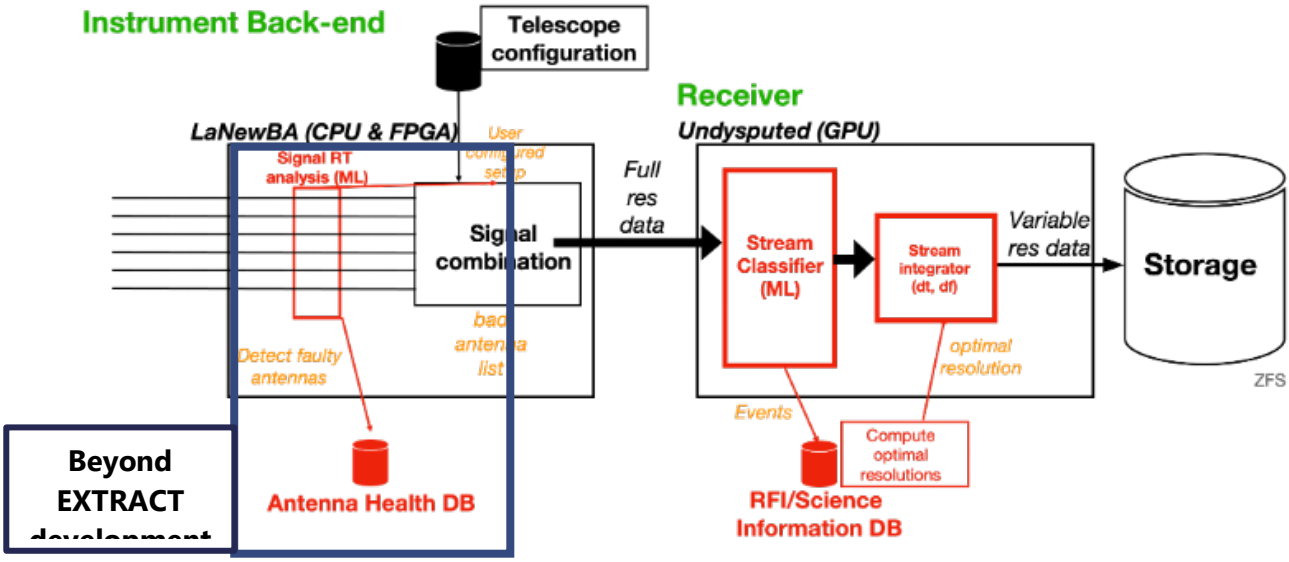

Figure 4.1: *Use-case A: (Right) Automatic event detector and classifier. The output of the classifier will be logged and also serve for decision-making in the stream integrator down-stream. (Left) Future development (after the end of EXTRACT) of a similar system can detect faulty antennas and decide to weight them down before signal combination.*

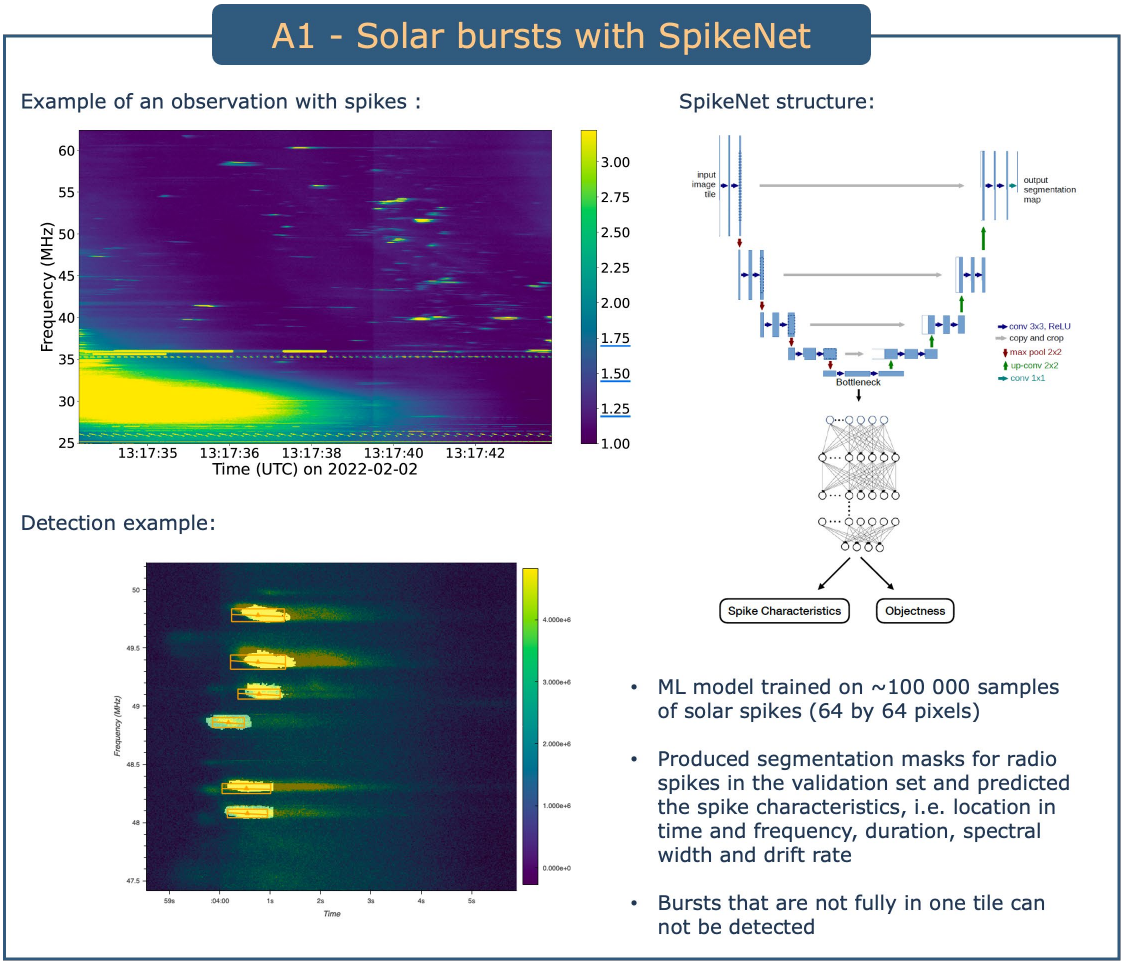

Figure 4.1.a : *(top left) Example of a solar observation containing interesting short spikes between 35-60 MHz. (bottom left) Positive detection of individual bursts. (top right) Structure of SpikeNet neural network, based on a U-NET.*

In use-case A1, the aim was to adapt a previously existing spike detection network in a real-time context. Fig. 4.1.a displays a small extract of the data flow (under the shape of a dynamic spectrum, frequency vs. time) showing large radio emission from the Sun superimposed with short-term spikes that need to be detected in time en frequency. Given the large amount of data, manual labelling and segmentation is no longer possible to be performed manually. However, it constitutes a really straightforward application for machine learning training. Based from a input data set of spectra and associated labels, a U-NET was successfully trained to quickly infer the presence and spatio-temporal location of the spikes (Fig. 4.1.a bottom left). With proper training, inference time is so short that it was integrated in the real-time acquisition system of NenuFAR.

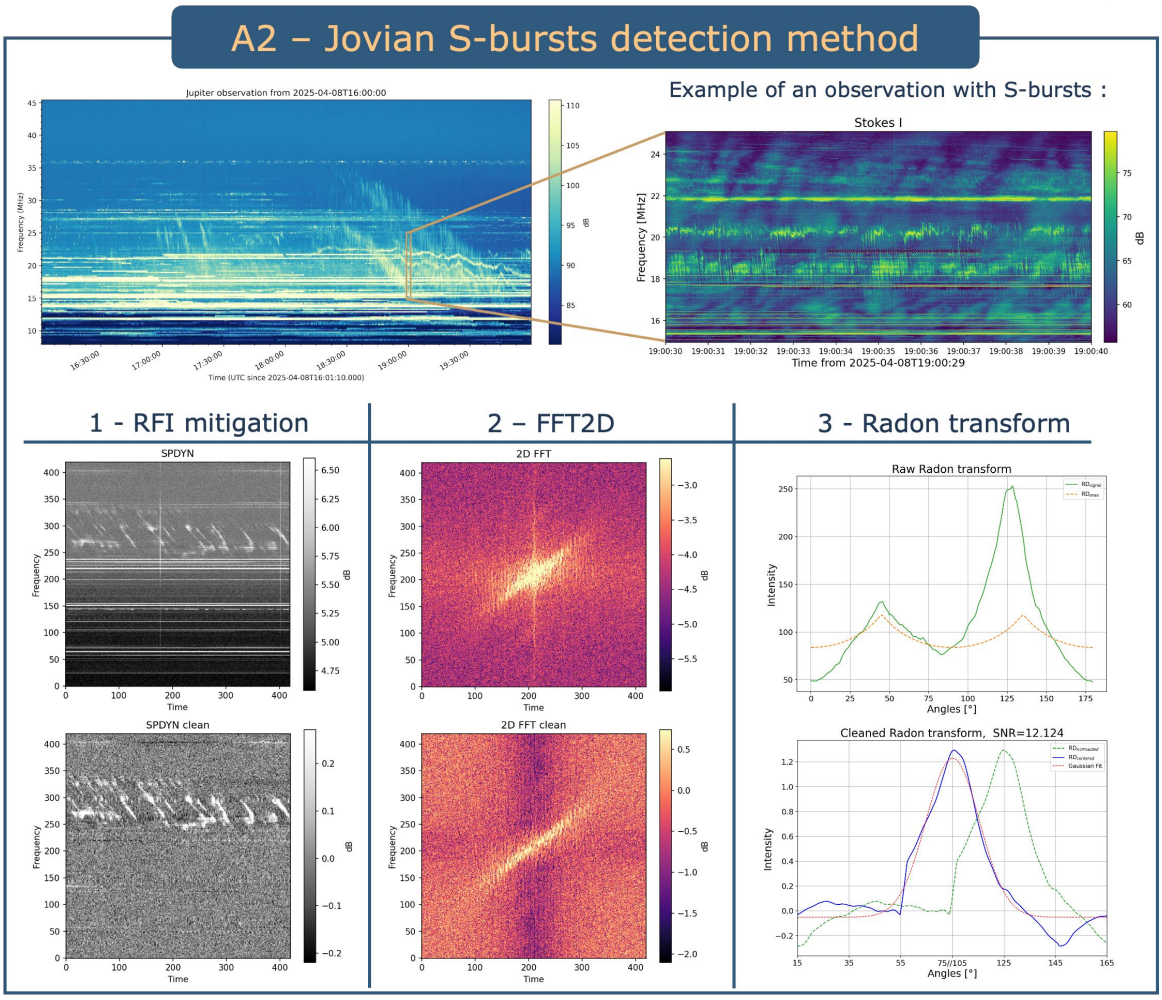

Figure 4.1.b : *(top left) 4-hour Jupiter observation between 10 to 45 MHz, showing slowly varying radio emissions. Horizontal features are radio frequency interference whereas the tilted cascading time-frequency features are radio emission from Jupiter. (top right) Zoom-in on 10 sec of observation showing numerous much shorter emissions (~10 msec) called millisecond bursts (S-bursts). (bottom from left to right) Zoom on a 1.1-sec section displaying a few S-bursts.*

In use-case A2, a detection algorithm was developped, not to detect and locate spikes but to detect and assess the morphology of short burst at Jupiter. Fig. 4.1.b, displays an example of each steps of the detection algorithm: 1) RFI mitigation : Thresholding is applied for RFI detectgion and removal, what is left after cleaning is dominated by the Jovian emissions. 2) 2D Fast-Fourier Transform used to detect the quasi-periodic drifting structures in the time-frequency plane. 3) Applying Radon transform (integration over a centered rotating line) on the FT spectrum in order to obtain a 1D-spectrum, with a peak at the angle corresponding to the drift of the S-bursts. This allows to then fit and identify, based on a signal-to-noise ration, the presence of S-bursts in the processed box and measure their drift rate.

Fig. 4.1.c to Fig. 4.1.e illustrated the adaptation work that was carried in order to integrate the real-time burst detector. From the existing data path Fig. 4.1.c, we took advantage of the presence of GPU and python implementation of GPU to insert the necessary stages that perform the inference of live data. This instrumental upgrade ran successfully and the specification were met and lead to a compression of 90% of the data volume, keeping the scientific content of the data intact.

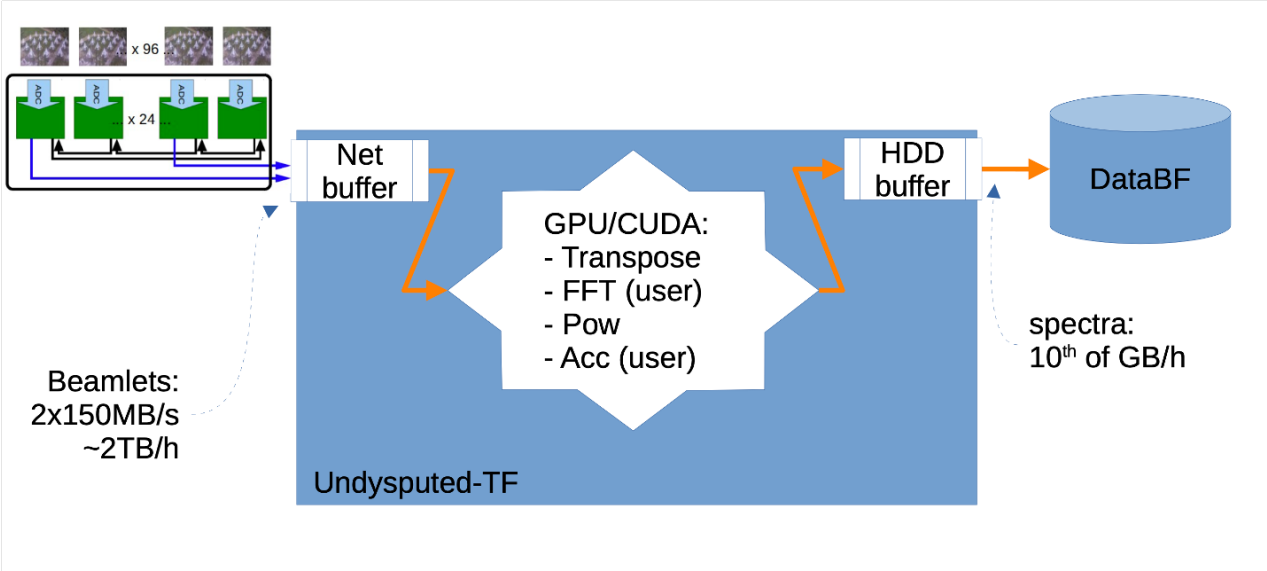

Figure 4.1.c : *Existing data flow going through the NenuFAR/UnDySPuTeD-TF interface.(from left to right) from the antenna Analog-to-Digital converters before entering UnDySPuTeD-TF when data are buffered, served to GPU through transposition, FFT, accumulation, ending up on hard drive.*

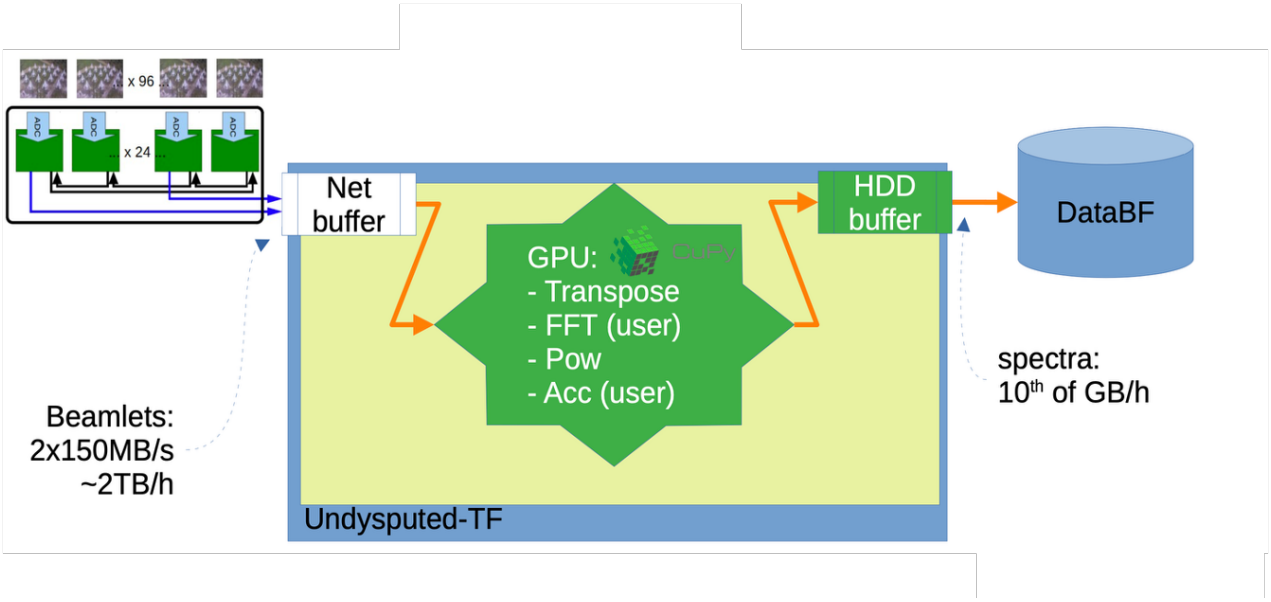

Figure 4.1.d : NenuFAR/UnDySPuTeD-TF interfaced with the MurMure framework written in CuPy.

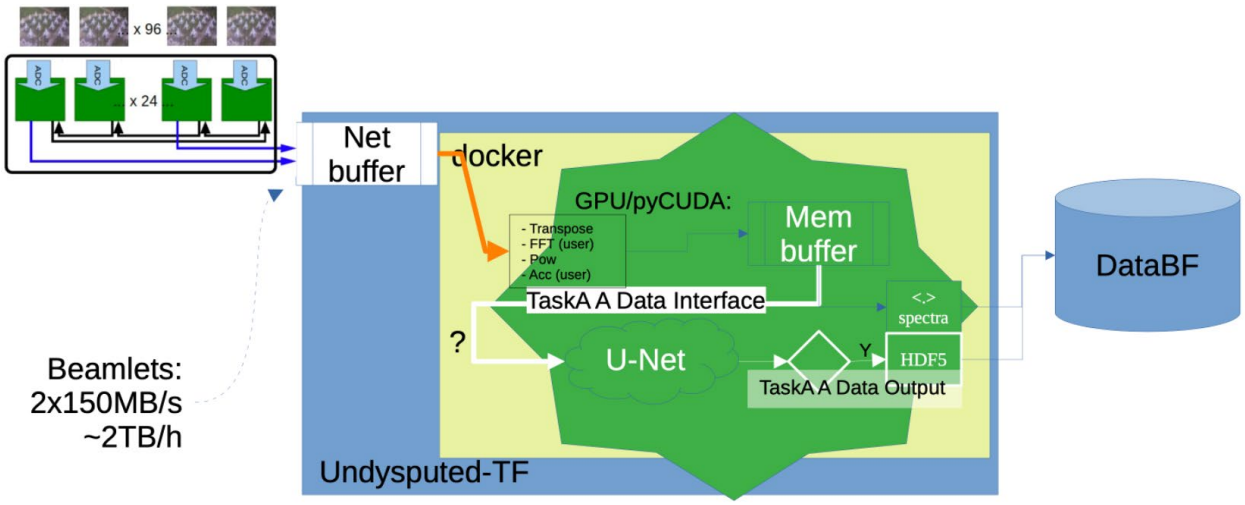

Figure 4.1.e : *Modification for integrating the EXTRACT TASKA solution (use-case A1) inside NenuFAR/UnDySPuTeD-TF interface with the MurMure framework written in CuPy. The trained network in directly performing the inference inside UnDySPuTeD connected to the memory buffer. Being placed between the memory and the accumulator step, decision-making is deciding dynamically time and frequency accumulation factors berfore recording to disk.*

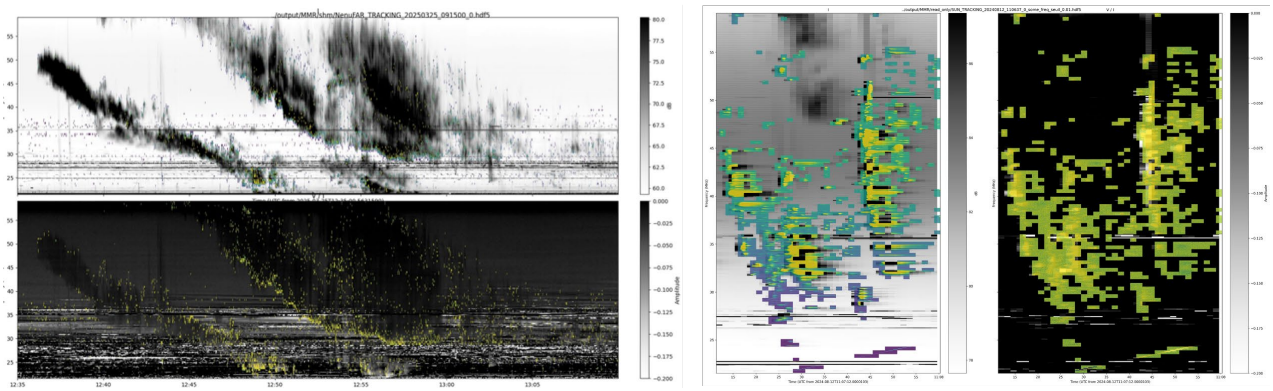

Figure 4.1.f : *(left) Clean spectrum showing Solar bursts (right) assignated labels after inference from the deep learning network.*

In conclusion, A1 was successfully applied to real-time detections in Solar observations with NenuFAR, resulting in a reduction of a factor 10 in high-resolution outputs (a paper is in preparation on this result), and A2 is on the verge of being ready for real-time detections in Jupiter observations. In the remaining period of the project, more focus will be put on the following the tasks :

- Producing a common and standardized output format for the detections (e.g. TFCat),
- building a labelled database for Jovian S-bursts, in order to train a neural network based on anomaly detections,
- optimize the parameters of the chain to enhance even more the data volume reduction factor.

4.1.2. Use-case C final specifications

Recall of use-case C motivation

The requirements for use-case C remain the same as described in D1.1 and D1.2 in creating a dynamic framework to enable a user (specifically a scientist) to define a scientific workflow from datasets coming from an instrument (edge). The application case is the reduction of radio interferometric data produced by NenuFAR. The data are issued by the edge part and is ingested with the relevant scientific and technical metadata in a dedicated storage. The data are handled with object storage and made available for processing by the workflow.

The workflow is composed of individual tasks that will either compress, calibrate or image the input data. The scientific products are produced all along the workflow in the form of intermediary products and end-products. They carry a high scientific value for the scientist. The life cycle of the data and the chain of tasks are properly logged to enable reproducibility of the results and to enable versioning.

The main contribution of TASKA-C is the ability to process data as abstract 'objects', treating them as a single logical entity regardless of their size, quantity, or formats. These files can be physically distributed over multiple sites but always seen as "virtually local" objects. As for the computing resources, data reduction tools can be parametrized and the relevant parameters are distributed to the (remote and heterogeneous) computing facility, The user expects to have a high-level vision over the workflow and a high level of control by not being cluttered with manual data logistics/credentials.

Updated specifications (based from D1.2 specifications)

A-Workflow control

The workflow should be able to work from end-to-end seamlessly with little or no human intervention with a properly given workflow description and set of task parameters required for the correct reduction of the data. The workflow should also enable for a way to interact with the workflow. Indeed, the user should be able to pause the workflow to inspect the intermediary products, make decisions, prepare parameters for the next tasks, and stop, resume, or restart tasks with other set of parameters.

B-Workflow data robustness, flexibility and upscaling in volume

Datasets are made available in the cloud environment as a collection of objects that can be accessed and processed by composing the tasks defined in the workflow. These datasets can vary widely in size—from hundreds of megabytes to several terabytes—so the workflow must minimize both data reduction time and energy consumption. Optimization strategies include efficient data partitioning, parallel processing, and dynamic decision-making for the optimal distribution of computational tasks across workers. Energy usage is further reduced by adjusting the workflow execution according to urgency levels, with lower urgency tasks consuming less energy.

C-Workflow user composition

User interaction includes building the workflow by arranging the task sequence, setting the object storage inputs/outputs, inspection of the final or intermediary data products & results and the use of a user script to control the logic of the workflow (e.g. looping

a set of tasks until some quality criterion is met). Users should be able to seamlessly access the various outputs. Users can interact with the workflow through a Python notebook, or a single Python script and possibly a dedicated interface for the design or accessing the results.

D-Data provenance

By following the principles of the IVOA provenance data model (https://www.ivoa.net/documents/ProvenanceDM/20200411/REC-ProvenanceDM-1.0-20200411.pdf), scientific activity involving data, numerical methods, codes, should respect the reproducibility principle by peers. Any peer should be able to access the minimal information to reproduce the result (which data processing, which code version, which data, which transformation, etc.). In order to respect these principles, the results should be attached together with the dataset as part of the meta-data. This should enable searchability of any data or products with respect to the relevant parameters (scientific or technical, such as the source direction in the sky, down to the date of last data processing). An illustration of the current implementation of the provenance, build along with the ingestion of data in the data catalog is illustrated below.

Final overall structure of use-case C

The figure below describes an updated view of the structure of the use-case C (adapted from D1.2). On the edge part, the data are produced and are put in a dedicated format (Measurement Sets, "MS" format, described in D1.2). One dataset is composed of multiple MS files that should undergo the same data reduction process in the workflow. To be able to define and set up the workflow, some mandatory metadata needs to be known by the system (see Fig. 4.1.2a). Data ingestion organises and store the dataset and produce the relevant meta-data. The user should use a predefined data reduction recipe or design its own workflow. COMPSS and Lithops (see D2.3, D3.3, D4.3) are the technologies that respectively form the creation of the workflow and the encapsulation of the software and the management of resources and partitioning. Metadata and provenance information are stored in the data catalog.

The list of available tasks (described in D1.2) can evolve and fit the current framework defined in the MVP in D1.2.

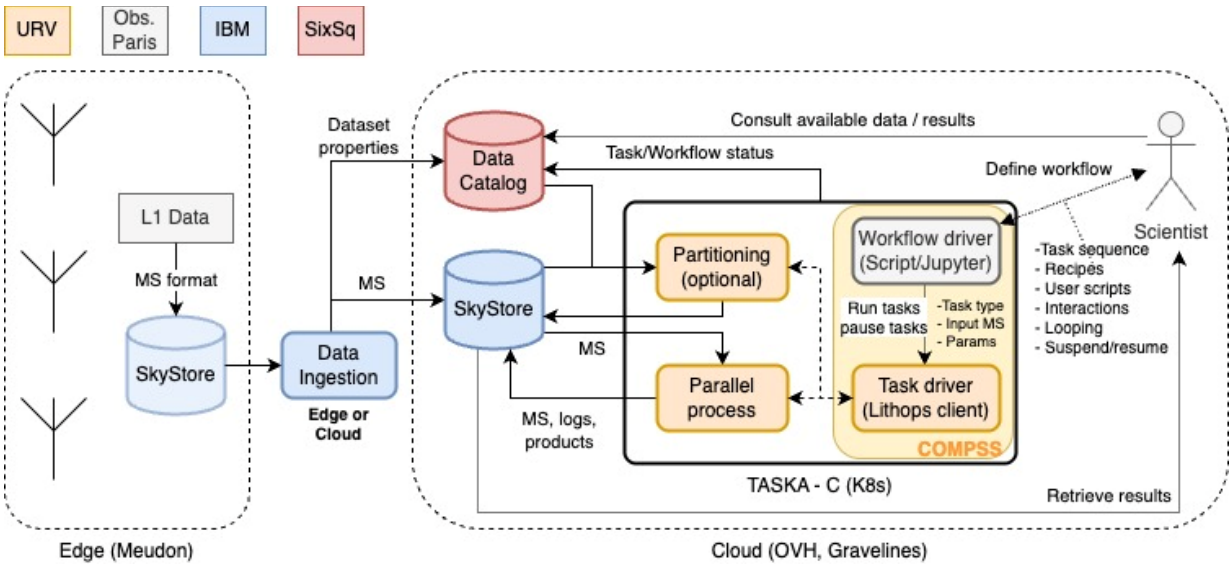

Figure 4.1.2a : *Main view of the edge/cloud interaction system for data reduction orchestration in use-case C*

4.1.3. use-case C - Data Management

Data ingestion of the cloud solution
Data ingestion consists of handling the data sets coming from the edge, extracting meta-data contained in it, and building complementary meta-data. This meta-data should be stored, be reachable and be searchable by the user for later use.
For example, the meta-data can consist of the date and direction of observation, data volume, scientist ID, etc.

The intermediary products (e.g. calibration tables, logs, image cubes, etc.) also need to be ingested as well as the input parameters that led to these products. Multiple runs of similar workflows should be logged with different workflow ID.

Logs are mandatory for the user to track down inadequate results and remember or inform third parties on how a product was generated, from which dataset, with which set of parameters. The ingestion of these extensive information should guarantee the scientific reproducibility of the products.

All the stored meta-data should be searchable by the user on a basis of keywords (e.g. target name, program name, etc), or ranges of values (e.g. time, frequency ranges, cone-search, etc.)

Available data set for processing and training
The dataset defined in D1.2 will serve as performance evaluation and includes small datasets, mid-size and large datasets to explore the parameters space and the various regimes of optimization (I/O limited, CPU limited, etc.).

These datasets vary in volume (from a few GBs (2.7) up to ~740 TB), in number of Measurement Sets (from 40 to 244) and in duration (from 10min to 238 min).
As described in D1.2, we add a large dataset (1.4 TB) from the NenuFAR Low Frequency survey that is purposely spread across multiple S3 object storage location (OVH, ODP and possibly Nancay). Finale application could help reduce the 49 TB of the whole NenuFAR low-frequency survey.

4.1.4. use-case C - Processing Tasks and Task execution

The ingested data will follow a route through the workflow and undergo various data processing tasks. In an effort to formulate generic tasks, the main required operation on the data resides in three main steps: i) data flagging & rebinning, ii) instrumental calibration and iii) data imaging. Most recipes can be decomposed in a combination of these tasks. Multiple instances of each task can occur for a given workflow (e.g. rebinning, then calibration, then a second rebinning calibration step) and the nature of the recipe may vary depending on the scientific objective and observation target. The system should enable various implementations and combinations of tasks in a flexible way. For the MVP, the flag/rebin tasks are covered by the DPPP software, while the imaging task is covered by WSCLEAN. In the final release, multiple imagers are available: WSClean, DDFacet (on HPC) as well as, Cassey, a home-made imager developed in the scope of use-case D.

- **Flagging/Rebinning Task (see. D1.2)**

Main purpose of the task
The main purpose of this task is to average down the data to filter out the fast and noisy variation as well as reducing the data volume.

Each Measurement Sets produced by the radio interferometer, contains raw data or pre-processed data in the edge side. However, further cleaning and compressing may be required before correcting for the instrumental response (i.e., Calibration task). The first task is a combination of flagging and rebinning. The first step consists of detecting outliers and bad samples by declaring them as "bad" inside the dataset by flipping the "FLAG" state of the concerned sample in the FLAG column of the MS. The subsequent tasks will be aware of the FLAG status of the samples and simply ignore this data and possibly reweight the data bins accounting for the missing "bad" data.

Main input parameters
The input parameters consist of the rebinning factors (in time and frequency) and the outliers' detections parameters (e.g. threshold levels, memory allocation, algorithm-specific parameters). The Flag/Rebin strategy depends on the nature of the observation.
Input MSs usually have a time integration interval (from 1s to 8s per sample) and number of frequency channels (from 64 channels to 1 channel) which define the rate at which the data were stored. Data flagging deploys outlier detection algorithm on these data matrix and decide which to choose based on multi-scale sliding window statistics. The rebinning factor can go from a factor of 1 to 12 in time, and 1 to 64 in frequency. These input parameters are specified in a DPPP parset file.

Data inputs/outputs
The flagged and rebinned data cannot be stored in the same measurement set. The high-resolution dataset will not be modified. However, the resulting measurement set will be written in a new MS file which will be smaller in volume, but will reproduce the overhead (subtables).

- **Calibration Task (see. D1.2)**

Main purpose of the task
The input dataset contains recorded information from the sky on an arbitrary scale. The data are also distorted by a series of external effects and instrumental effects. The purpose of the calibration task is to correct for these effects and scale the data on a physical scale before scientific exploitation of the data.

Main input parameters
The calibration task requires a series of input parameters that are describing the way the data will be corrected for. These input parameters are specified in a DPPP parset file.

Data inputs/outputs
The input are the flagged/rebinned MS files as well as an external "sky model" (supplied by the user). The latter is mandatory to serve as a reference to deduce the instrumental parameters.
The default output is a series of calibration coefficients that will be applied to the dataset to produce the corrected MS. This series of coefficients (Calibration Table) are

stored in an HDF5 file format that can be stored, used immediately or applied to another set of MSs.
The calibration coefficients can be directly applied to the input MS and create a new column inside the current MS that is processed. The output data is the same as the input but with an augmentation of the table by the CORRECTED_DATA column.

- **Imaging Task: (see. D1.2)**

Main purpose of the task
The imaging step takes the rebinned/flagged and corrected data from previous steps and aims at inverting the data from the Fourier (visibility) space to the image (direct) space. WSCLEAN will load all data in memory and perform a gridding of the data on a regular grid. Then will perform a deconvolution step to cope for the missing of data.

Input parameters
The imager takes a series of input parameters (see Table 1) to define the gridding, the image size and the deconvolution parameters. This is subject to vary depending on the observation and depending on the scientific product that the user wants. Multiple instances of imaging can be run on the same dataset, in the search of finding the correct representation and deconvolution of the data.

Data inputs/outputs
The inputs are the calibrated MS and the outputs are image cubes whose dimensions depend on the imaging parameters.

- **Advanced Imaging Task: (Newly introduced task)**

Main purpose of the task
The simple imaging step can sometimes no longer suffice to properly render images. Advanced imagers (such as DDFacet or that developed in use-case D) are using a lot of resources and should be run in an HPC environment. The input/output of this step are the same as those of the classical imaging, however, an external call to an HPC platform is sent (taking the shape of a SLURM command) as well as the data mounted/transferred on the HPC side from S3 storage.

Input parameters
The imager takes a series of input parameters (see Table 1 in D1.2) to define the gridding, the image size and the deconvolution parameters. This is subject to vary depending on the observation and depending on the scientific product that the user wants. Multiple instances of imaging can be run on the same dataset, in the search of finding the correct representation and deconvolution of the data.

Data inputs/outputs
The inputs are the calibrated MS and the outputs are image cubes whose dimensions depend on the imaging parameters.

- **HPC Task (Newly introduced task)**

Main purpose of the task
Most task occurs on a kubernetes cluster however, for some tasks, HPC platforms are necessary. This motivates the creation on an HPC tasks which is able to trigger the chains of command

Input parameters
HPC settings using the SLURM format.

Data inputs/outputs

Since the computation is done on an HPC, the order is to ship the data where the computational efforts are taken.

4.1.5. use-case C – Example of a complex workflow and Provenance

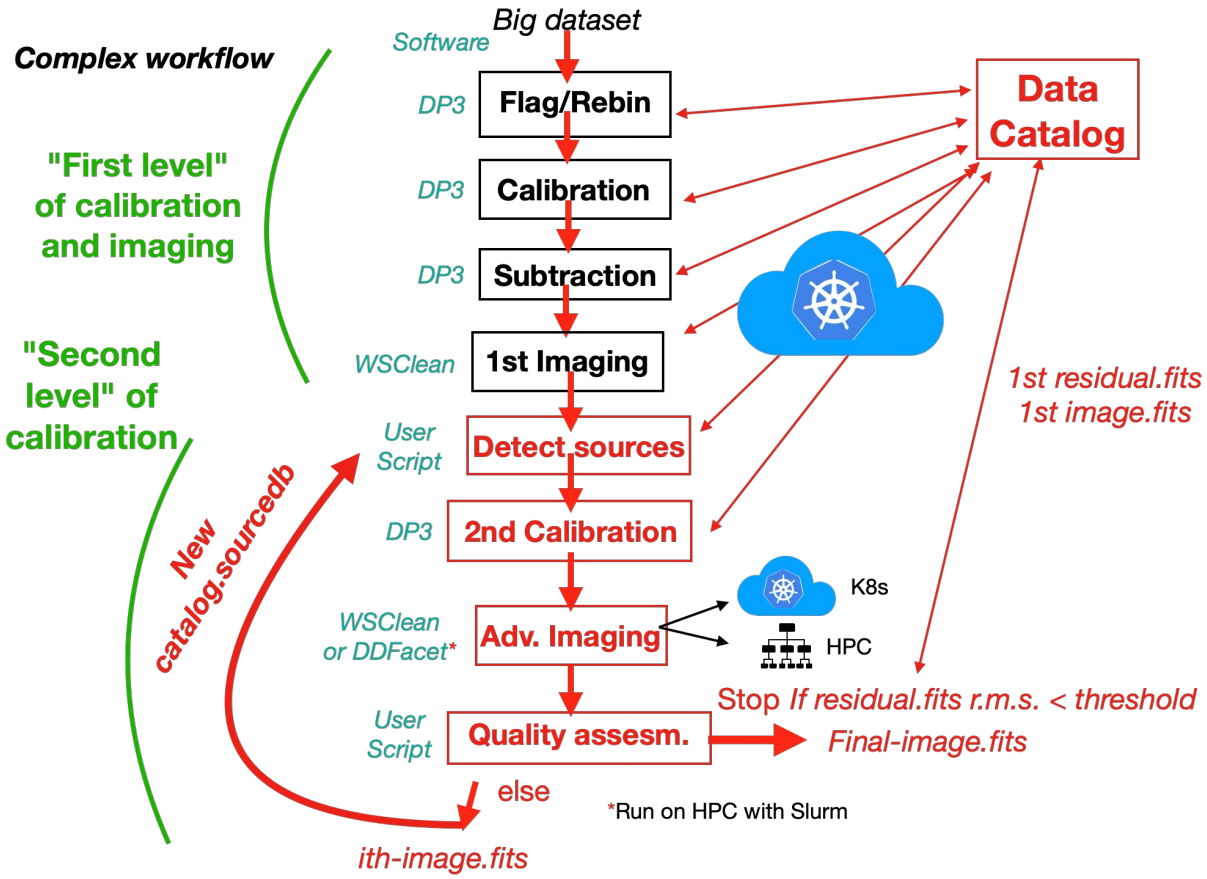

Figure 4.1.5a : *illustrates a realistic complex workflow (not showing the workflow branching between calibrator data and target data) taking raw data and producing refined images after undergoing flagging, rebinning, calibration, source subtraction, simple and advanced imaging and quality assessment. Part of the workload is performed in the Cloud on Kubernetes clusters, advanced task can be performed either on Kubernetes or in an HPC environment.*

Based from the past experience using the MVP (see D1.2) we have built more complex workflows to get closer to a more realistic workflow use on a daily basis by radio astronomer.

Multiples inputs and outputs are then chained and propose three new features: i) the capability to pilot Slurm jobs on a remote HPC infrastructure as a dedicated task in the workflow (see previous section), ii) the capability to compose the workflow that include task looping and logic decision-making (see next section) which need to be fully controlled by the orchestrator, iii) the capability to interact with the user during the workflow run by suspending the workflow, asking inputs from the user (e.g. STOP/CONTINUE) based on some triggering criterion (quality, maximum numbers of loops, stopping criterion met, etc.) and iv) data ingestion, data catalog and "Provenance" implementation for reproducibility.

These four features have been implemented (only partially for some due to the dependency to the performance assessment) and need to be tested for robustness and performance. The communication solution between the workflow and the user can be

implemented using multiple solutions such as ssh commands, MQTT (Message Queuing Telemetry Transport) interactions, etc. The final practical solution will be chosen at the end of the evaluation period.

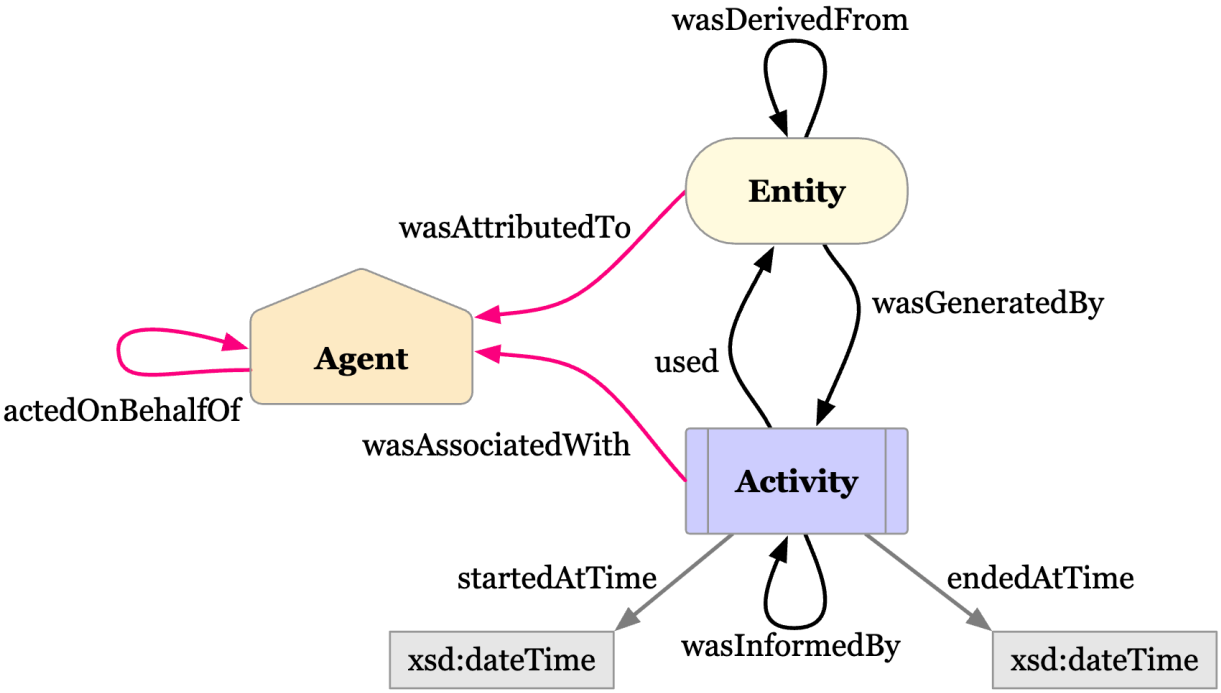

Figure 1. The three Starting Point classes and the properties that relate them.
The diagrams in this document depict Entities as yellow ovals,
Activities as blue rectangles, and Agents as orange pentagons.
The responsibility properties are shown in pink.

Figure 4.1.5b : *Principle of the Provenance convention and logical link between Entities, Activities and Agent (from IVOA Provenance model V1.0 specifications).*

The production of reliable scientific lies on the capability to reproduce results with transparent tools (especially for peer reviewing and to reach scientific consensus). By construction, a strong feature such as the implementation of Provenance principles must guide the data reduction process. For any resulting products, anyone should be able to identify, which workflow run, which tools version, which data, which set of transformation have been applied to the data that resulted in the product.

Provenance principles (illustrated in Fig. 4.1.5b) enables the implementation of a representation graph by creating dependencies and relational links between data, processes and agents. By following these standards, we contribute in the FAIR principle (see https://www.ccsd.cnrs.fr/en/fair-guidelines/) which guarantee the data and products to be "Findable, Accessible, Interoperable, Reusable".

The natural place to insert Provenance "grammar" and rules are in the data catalog (using Nuvla technologies) along with input data metadate generated during data ingestion. The data catalog serves as a "logbook" of everything that happen to data. From a given product, we have the information of the immediatly preceding action that led to the existence of the product. By moving upward in the chain from neighbours to neighbours, every product has a unique time line along which all used parameters and data are known in a deterministic way. It is critical to have this feature in the workflow since any change of version of any tool can easily be tracked and reduced ambiguity leading to unreproducible results.

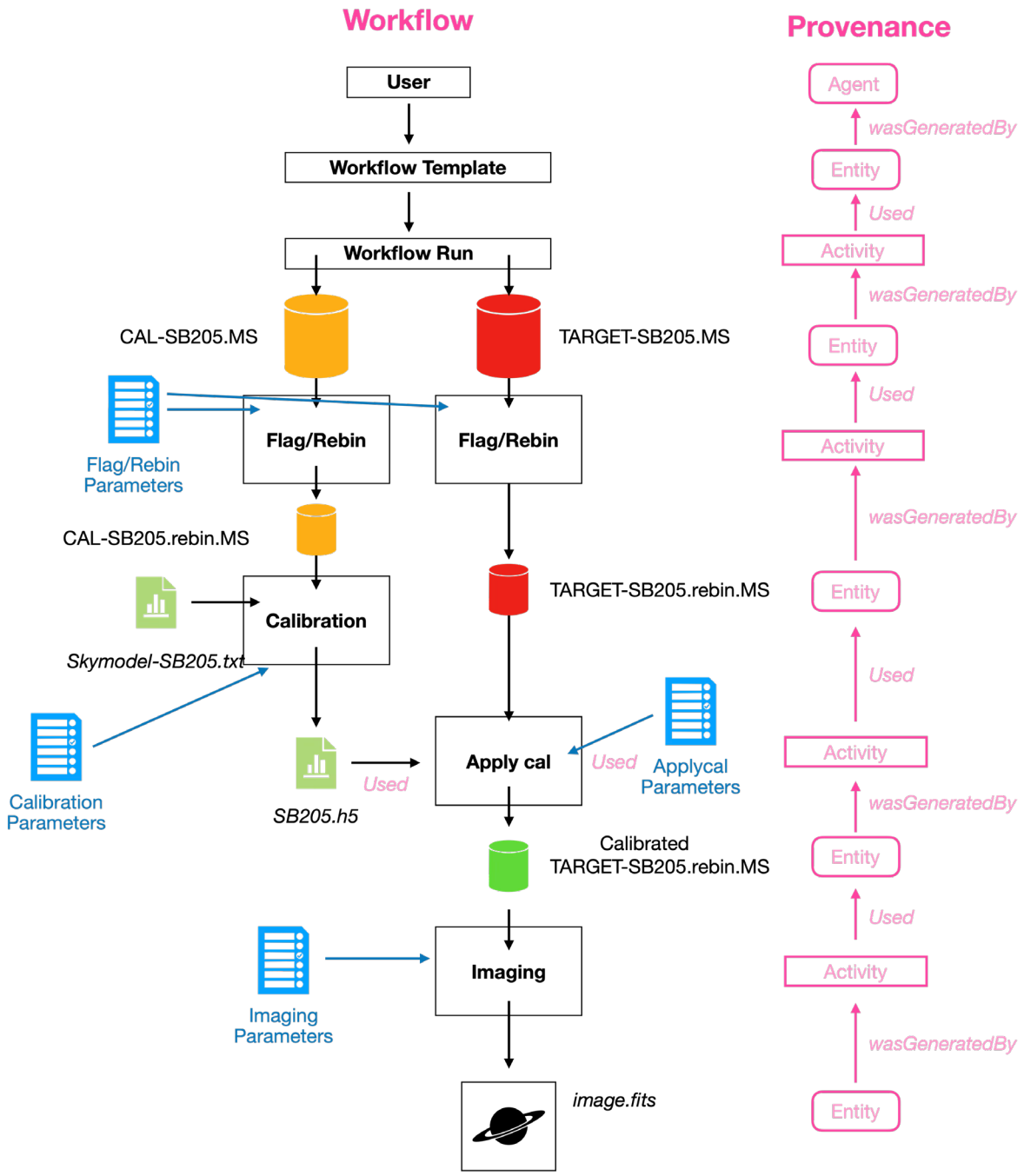

Figure 4.1.5c : *Illustration of the simple workflow (see D1.2) with corresponding Provenance graph (in pink). For each products or steps, it will be able to trace the history which led to any data product.*

Fig. 4.1.5c illustrates the implementation of the simple workflow of the MVP (see D2.1) in parallel to the translation into Provenance grammar and rules setting up the dependenvies between tasks and data. The user first select a model workflow template and create a new run instance of the workflow by running it along with a dataset and priori information. This dataset contains reference data ("calibrator" data) and the science target data "target" data). Each dataset follow the same first step "Flag/Rebin" before branching into "calibration" and "Apply calibration" steps. Final step involved imaging only the calibrated science target data. Blue parameters are specified in the workflow template as sets of parameters for each individual tasks.

4.1.6. use-case C - Workflow Management

In the development of the MVP, a critical aspect is the workflow definition which creates the sequence of computational steps required to process data effectively. The definition of a workflow dictates the systematic progression of data through various processing stages as well as leveraging advanced capabilities of distributed computing resources to enhance performance. This workflow leverages distributed computing resources to enhance performance, aligning with the TASKA use-case goals of a seamless end-to-end processing and minimal human intervention.

Interactive workflow definition

We have used Python for workflow definition. This allows workflows to be programmed in simple scripts. In particular, Jupyter notebooks serve as an interactive playground to define workflows. This integration provides a user-friendly interface for defining them. Through these notebooks, scientists can iteratively build, test and refine their workflows, visually monitoring the steps' outcomes and making adjustments in real-time.

The workflow is composed by combining well-defined components that execute and scale in the cloud. Each component is called a step and they act as a function: they receive an input (e.g., a measurement set, or parameters) and produce one or more outputs (e.g., a modified measurement set and/or the operation subproducts). Parameters for a given step can be dynamically adjusted based on the outcomes of the preceding ones, allowing the workflow to adapt and evolve based on the user/scientist's requirements.

The interactive mode of composition is optimal for designing new workflow and test it seamlessly. In the final release, it is possible to define the workflow as a stand-alone python script describing each step of the workflow with decorators that enable COMPSS (see D2.3 and D3.3). Defining chains of tasks including looping capabilities.

```
267 if __name__ == "__main__":
268     print("Starting workflow")
269     # Orchestrate step execution
270     calibrator_rebinning_output = CAL_rebinning()
271     calibrator_calibration_output = CAL_calibration(calibrator_rebinning_output)
272     target_rebinning_output = TARGET_rebinning()
273     target_calibration_output = TARGET_calibration(
274         target_rebinning_output, calibrator_calibration_output
275     )
276 
277     Niter=0
278     Stop=5
279     Imagestepout=[]
280     while(Niter < Stop): # criterion
281         print('Entering Imaging #%s'%str(Niter))
282         TARGET_imaging_params[-1]=OutputS3(bucket=BUCKET, key=prepend_hash_to_key("TAR/imag_out/"), file_name="image%s"%str(Niter))
283         if Niter==0:  # first imaging depends on the output of the calibration
284             Imagestepout.append(TARGET_imaging(target_calibration_output,TARGET_imaging_params))
285         else:  # Niter is 1 in minimum
286             Imagestepout.append(TARGET_imaging(Imagestepout[Niter-1],TARGET_imaging_params))
287 
288         Niter+=1
```

Figure 4.1.6a : *Main section of the workflow script with COMPSS. Building a loop composed of workflow steps is made seamless thanks to the python interface Lithops and COMPSS.*

Workflow interactions

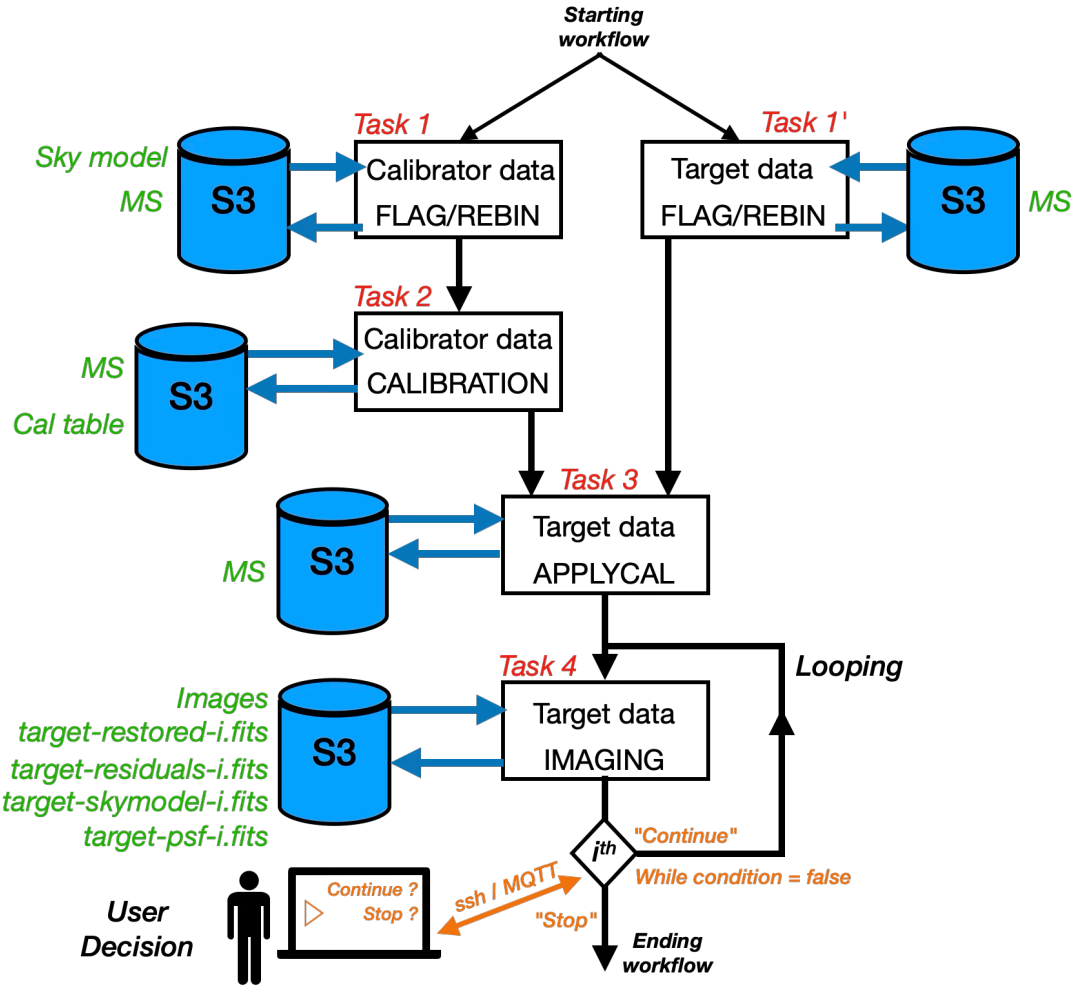

Figure 4.1.6b : *Example of simple branching and looping workflow. Scientific data are process in parallel to the calibration data. Task 3 depends on task 2 and task 1' before moving to the next task. Looping is possible through scripting and user agency is also possible for interacting with the workflow (STOP, CONTINUE).*

In addition to the MVP implementation (see D2.1), we implemented the interaction with the user, not only for composing and executing the workflow, but also to introduce a semi-automatic level of control where the user is asked to react upon a logical question occuring during the workflow run. For example, if a task is looped based on some quality criterion or a number of iterations, the workflow should be suspended (not killed) waiting for the user input to continue or stop the workflow.

This unlock the capabilty to have complex workflows in which the user can intervene to assess quality, run a specific user script on the intermediary products or simply validate the quality of the accomplished tasks as a matter of control (in case of a task run of heavy data, it is sometimes preferable to have a human intervention in the middle of a ressource heavy workflow run, in order to avoid mistakes and minimize potential ressource waste).

Step Implementation and Parameter Management

To code this composition in simple Jupyter notebooks, we developed a light Python library. Each step is encapsulated as a class that inherits from a base PipelineStep, which defines a standardised structure for implementing diverse processing tasks (the different steps, see Figure 7). This follows the objective to have a flexible task implementation and execution. This structure includes methods for executing the step's core logic, handling inputs and outputs and defining any necessary parameters that control its behaviour. This way, we create a modular and reusable system where each step can be independently developed, tested and integrated into workflows. It is also extensible, allowing to add other steps or different implementations in the future.

Parameter management is linked with step implementation, providing a mechanism to adjust the behaviour of each step dynamically based on the workflow's needs and intermediate results. Parameters for each step may include configuration settings, thresholds for data processing and paths to input/output data. These parameters are provided to the step in a dictionary data structure and may be defined and modified interactively, leveraging the capabilities of Python and Jupyter notebooks.

To create workflows, scientists combine implementation steps and parameter management as needed for their scientific objectives. This allows them to define complex workflows that are both adaptable and scalable across cloud environments.

Figure 8 is an example of how the workflow is defined and how it is executed. We see that the workflow definition consists of inputs and outputs. The input is a path to where the data is located, which can be a single measurement set or a collection of them, while the output is where we want the resulting rebinned measurement set to be stored for the next operations.

Internally, this workflow execution leverages the EXTRACT platform to run these steps in the continuum. In the MVP, the computation was deployed in the cloud and managed by the Lithops data-driven resource scaler. The run method takes the step's parameters and calls the Lithops library appropriately to enable scalable and cloud-based execution, taking advantage of distributed computing resources. It will create as many parallel workers as needed for the input data chunks we have defined and will output the corresponding products after the operation is applied. In the final release, both the "interactive" and "non-interactive" workflow design capabilities are possible. Most of the effort in the remaining period will focus on implementing new tasks and optimizing the computational/storage resources by playing on parallelism/partitionning strategy that may vary depending on the dataset. The dataset described in D1.2 will be used for assessing final performances.

4.1.7. use-case D final specifications

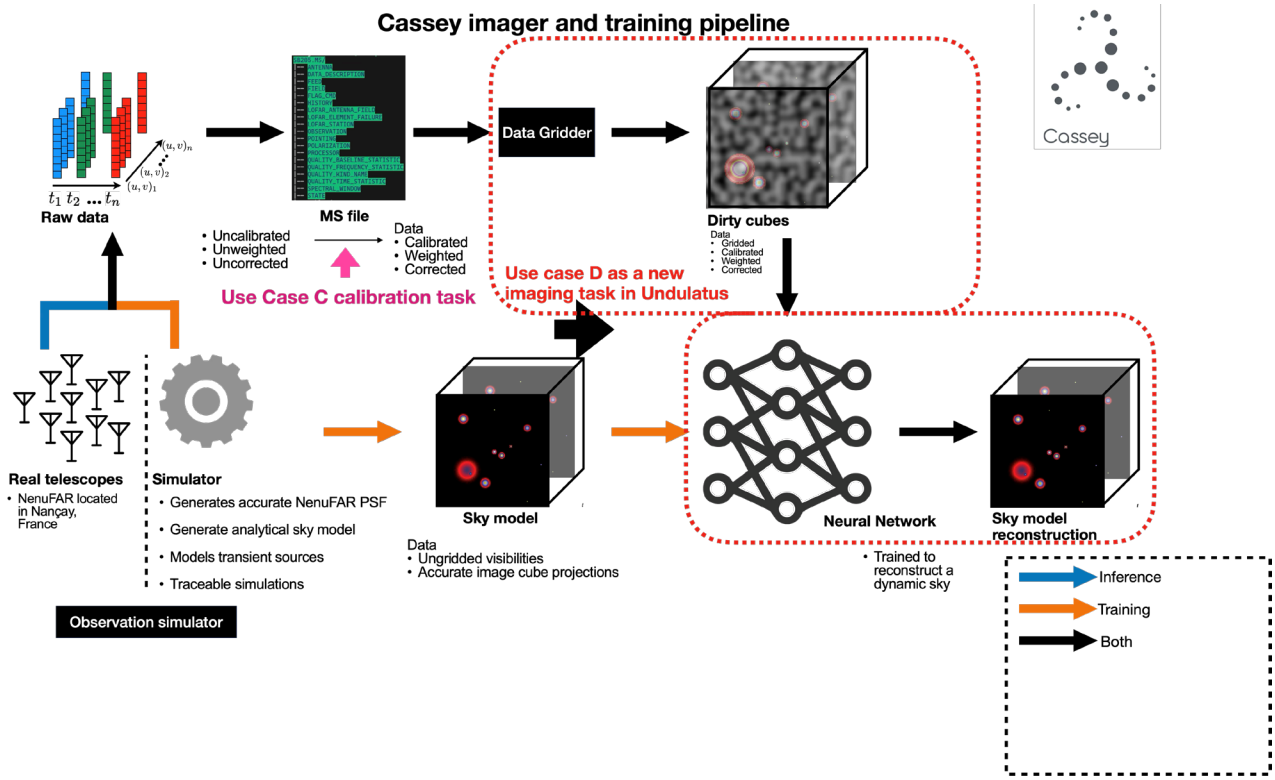

Figure 4.1.3a : *General overview of use-case D structure. Signal is generated by real telescopes or a telescope simulator (PSF generation, sampled analytical sky model, variable sources with multiple light curves, observational parameters) and go through the processing*

A new dedicated imager has been proposed in D1.2 to replace the current imaging step tool (WSClean) which produces regular images cubes from data containing static sources. However, when the data contains variable sources, the current imager is unfit for proper imaging and reconstruction of the light curve. The issue is critical in case of a moving emission (e.g. Starlink reflections distorting the data) as well as "extended" moving emission (e.g. Solar CME radio emission, the main scientific driver of the TASKA use-case). To address the classical imager limitation, we devised a new deep learning network which integrates the variable nature of the source and which is trained on representative data containing a mixture of static sources and variable sources following typical light curves (based on Deflation net network, proposed in Chiche et al., 2023). Recent development in Deep learning network structure and application to video restoration is an inspiration for solving the inverse problem of imaging. Transformers (as proposed Akhaury et al, 2024) are good candidates since they are aligned with the incomplete nature of the visibility measurements. On this basis, we present in Fig. 4.1.3a, the final architecture of the new imager. Once trained and validated during the final period, it will be seamlessly integrated as a new task in the workflow of use-case C. use-case C was indeed prepare to host -any- tool that can be deployable in a Docker container, see D2.3, D3.3. Calibrated data produced by Undulatus (use-case C) enters the imager and are prepared for inference. Trained coefficients are stored in the model serving module (or uploaded on S3 object storage for customised training by an advanced user) and go through the trained network. Outputs of the network are the restored, deconvolved data cube containing static and variable source, as well as error estimated on the quality of the reconstruction. The training of the network required us to develop an observation simulator (described

below) as an original contribution to the project, as it is critical for a realistic training of the network. The observation simulator uses the real data format and inject realistic time-dependent sky models and instrumental response. The simulator itself is highly valuable by-product of the development which goes beyond EXTRACT as it can serve as an independent package to simulate -any- radio telescopes data (given the telescope constant is inserted in the code).

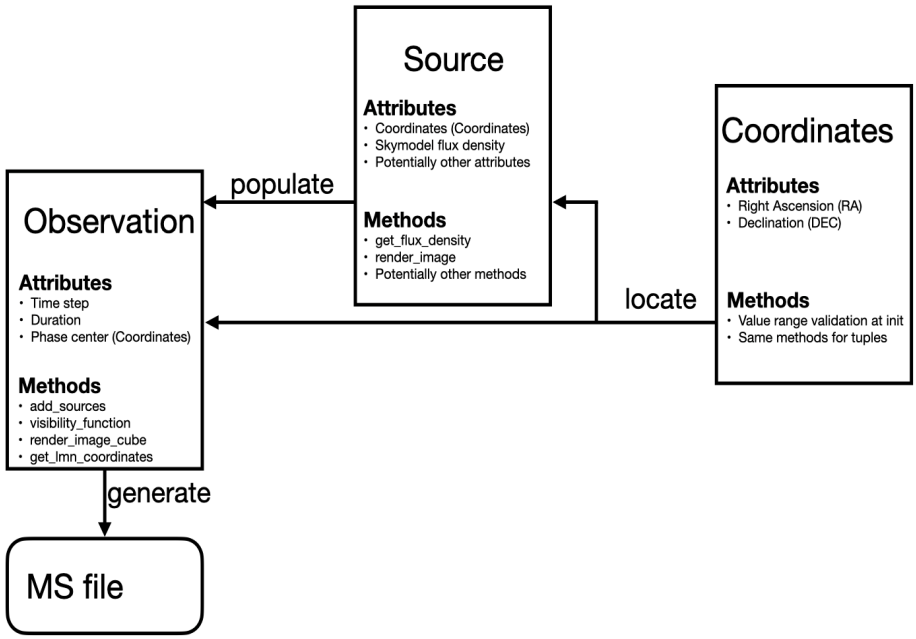

Figure 4.1.3b : *General overview of use-case D observation simulator. An Observation instance is able to generate a MS file. The Observation instance is populated by Source instances. Observation and Source instances use Coordinates Class as type of their attributes.*

The simulator module in allows to generate realistic MS files using real telescope responses separated in three main blocks (see Fig. 4.1.3b). The file is obtained by sampling an Observation instance using a real telescope configuration. The Observation is populated with Source instances. The implemented source subclasses correspond to Point Sources that are either constant or transient. Transient Point Source have 4 types of temporal profiles: Step, Gaussian and Skewed.

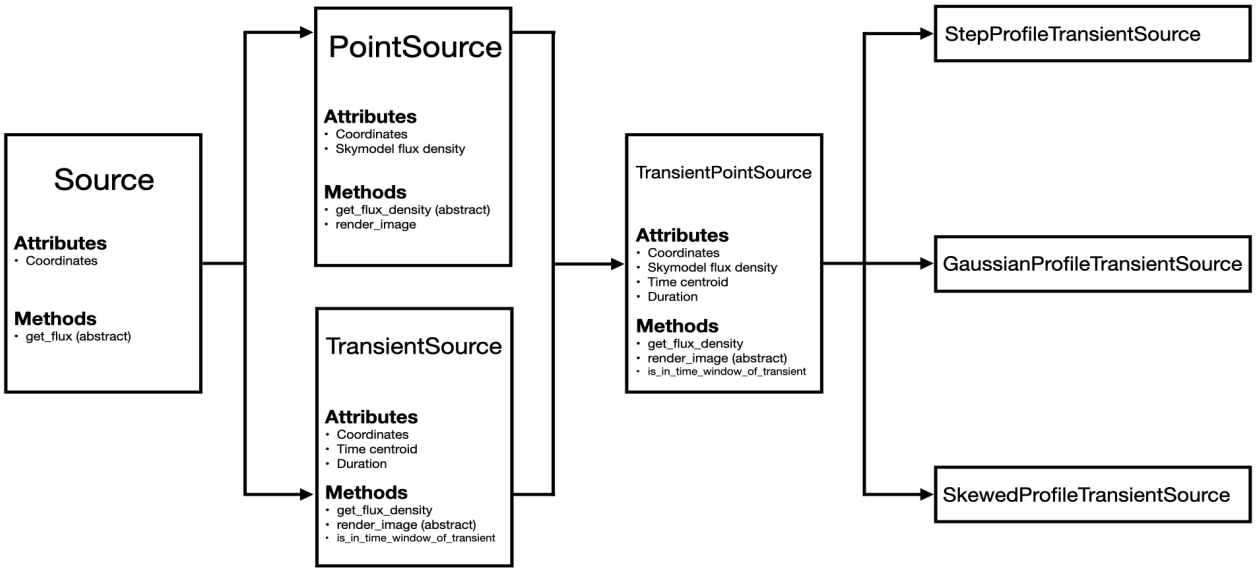

Figure 4.1.3c : *General overview of use-case D Source class. Different transient profiles only differ with their get_flux_density method implementations. Their are all child classes of TransientPointSource which has diamond inheritance of Source and its two child classes Point Source and Transient Source*

The Source class has four subclasses that are directly used in simulation (see Fig. 4.1.3c): PointSource, StepProfileTransientPointSource, GaussianProfileTransientPointSource, SkewedProfileTransientPointSource. Using object oriented programming principles, Source, TransientSource and TransientPointSource are abstract classes that allow a modular, extensible and maintainable implementation.

The inference pipeline and the training pipeline share modules in common. The inference pipeline consists of receiving astronomical raw data from radio telescopes then calibrating and correcting it inside a MS file (done as previous steps of use-case C). Then in the imaging step, the gridder constructs a dirty image from the MS file and the neural network restores the image and reconstructs a sky model. In the training pipeline, instead of receiving raw data from telescopes, it is generated by the simulator that also outputs the exact corresponding sky model which is used for a supervised training of the network.

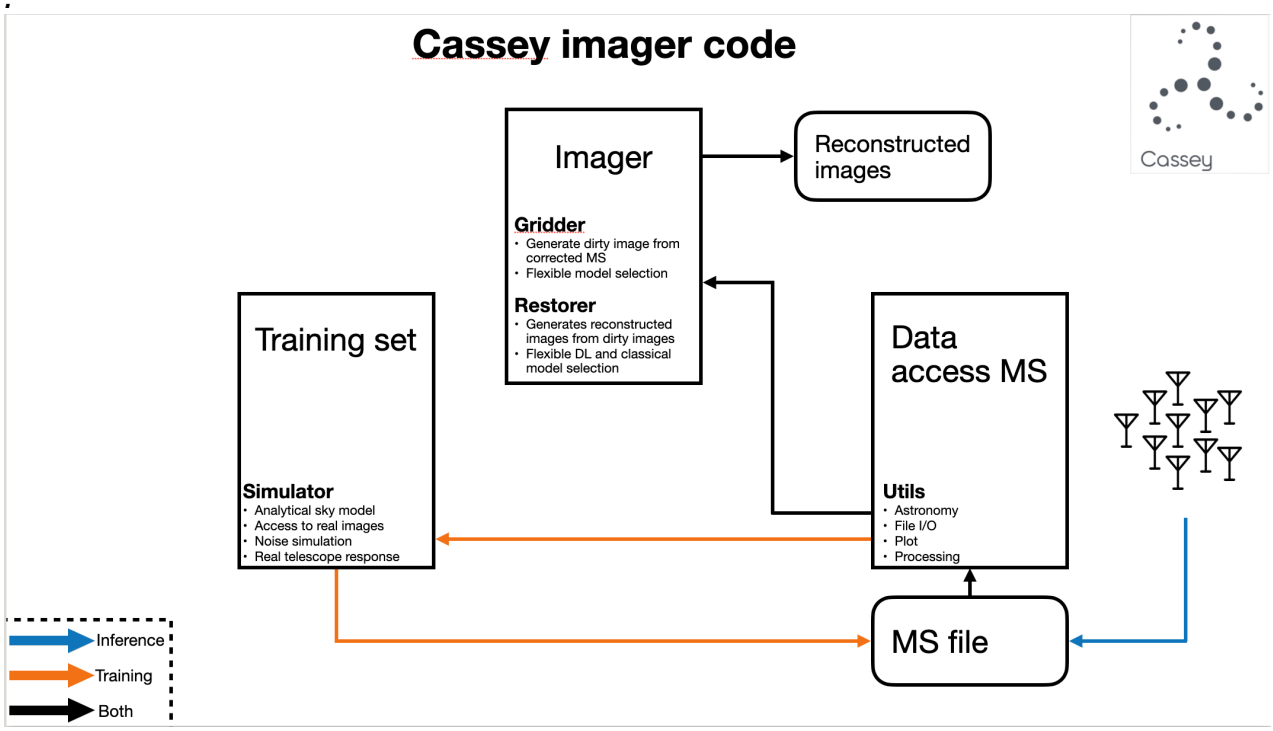

Figure 4.1.3d : *Structure of the new imager taking MS file format as input, going through a data access library to serve the imager. This imager contains the trained network which does the inference and the reconstruction of the image cubes. An observation simulator has been developed along with the imager, in order to prepare the training, validation and test data sets.*

The imager code is structured into a group of three repositories (as depicted in Fig. 4.1.3d and available on the use-case-D project on gitlab):

- "Data access MS": contains utils scripts that are common to 'Training set'. One of the scripts is for the handling of MS files.
- "Training set": contains the scripts related to training. Including the data simulation step.
- "Imager": contains processing code to reconstruct sky models using raw data.

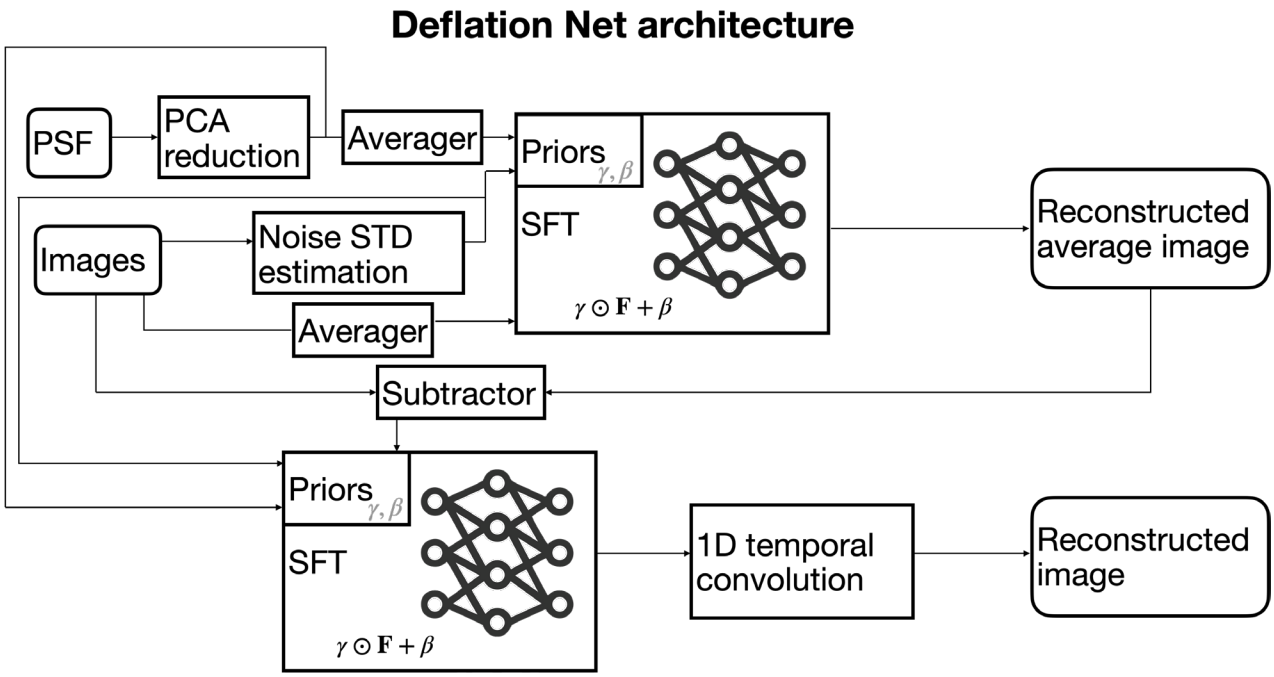

Figure 4.1.3e : *Internal block description of the Neural Network (Deflation Net). From the knowledge of the PSF and input data set, an average image is computed and subtracted to the data set. Variable source are then captured by the network and forms output deconvolved and denoise images. Restored light curves are then compare to the ground truth and reconstruction by the Clean algorithm.*

The adaptation of the Deflation Net (from Chiche et al., 2023) starts by ingesting the PSFs and the radio source images (see Fig. 4.1.3e). The PSFs are then reduced using PCA. The first step consists of reconstructing the average image by inputting the average reduced PSF and the average image into the SFT module. The obtained image should correspond to the constant sources in the observation. Then the reconstructed average image is subtracted from each image and the difference alongside with the corresponding reduced PSF are passed to the SFT module for reconstruction. The obtained images are passed through a learned 1D temporal convolutional network to improve each reconstruction using their shared temporal structure.

4.2. PER Final Release

4.2.1. Overview of the PER Architecture

The PER use-case within the EXTRACT framework addresses emergency evacuation in complex urban environments, with Venice as a representative scenario due to its high population density, constrained infrastructure, and geographic fragmentation. Building on the MVP described in the *Deliverable D1.2*, the current release reflects the final integrated architecture, including refined technological components, platform interoperability, and readiness for real-world piloting. As shown in **Figure 5**, the PER system integrates a **semantic UDT** that models the city using both historical and dynamic data. This enables real-time awareness of environmental and population conditions, forming the basis for scenario simulation and decision-making.

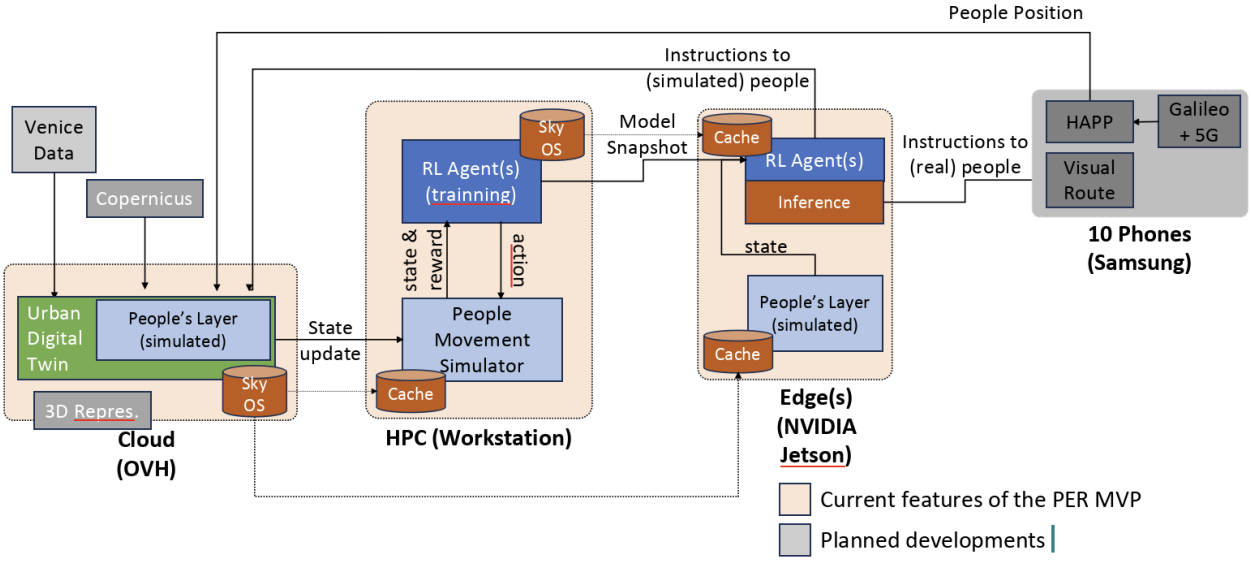

Figure 5: *PER use-case Architecture*

A central element of the architecture is the **Simulation Environment (SIM)**, which replicates individual and crowd behavior during emergency events. SIM serves a dual role: validating evacuation strategies under realistic constraints and triggering inference. This is achieved by passing event and state data from the UDT to the RL module. This enables the **closed-loop operation** of the PER system. Lastly, the **Data Layer**, and in particular the **Data Mining Framework (DMF)**, is used to coordinate data processing across the various flows comprising PER. Each dataset, whether introduced from an external source or generated within the system, undergoes ingestion. This includes registration of its location in the data catalog, which in turn triggers a notification for all subscribed consumers—allowing them to locate the data and use it as input for further processing. For example, both the simulator and the RL model inference receive updated city state information from the UDT, using object storage for data storage and the DMF catalog for data notification.

Another important component of the DMF is the **inference service**, which enables the RL model to operate *as-a-service*. This allows it to be accessed from one or more edge clusters—at scale if needed—using **KServe**. Further technical details on the DMF and Data Layer are available in *Deliverables D2.2* and *D2.3*. A significant advancement made during the last period is the refinement of the PER MVP to align more closely with the envisioned architecture of a real-world product, as illustrated in **Figure 6**. Most notably, this includes the introduction of a **5G MEC newly introduced component** (currently simulated by a service prototype) that resides at the edge and communicates bi-directionally with external devices located in the hazard area. These may include both real-world devices (e.g., smartphones) and simulated ones.

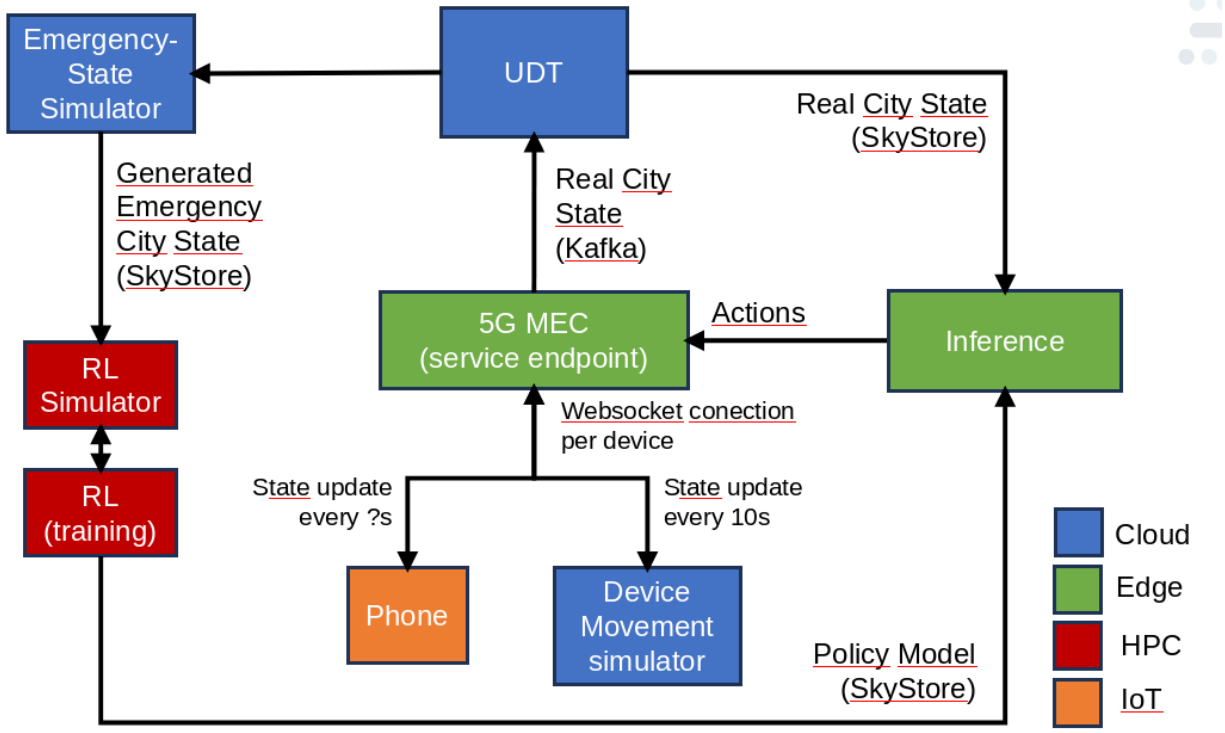

Figure 6: *Refined implementation of PER use-case*

The 5G MEC component serves several key functions:

- It collects information about external devices—both real and simulated—and propagates it to the UDT;
- It periodically queries the inference service for updated instructions covering the entire scenario (i.e., for all devices), and caches the results;
- It processes the cached inference output to generate **device-specific instructions**, directing each person to safety;
- These personalized instructions are then delivered to users via the **Evacuation Mobile App (EMA) newly introduced component**, which runs on smartphones and other connected devices. The EMA acts as the primary user interface, ensuring that individuals receive timely, context-aware guidance throughout the evacuation process.

The refinement work presented in Figure 10 concludes the PER MVP development and sets the groundwork for subsequent productization phases.

These will include user interface enhancements and the implementation of real-time communication features. The following sections of this chapter provide a detailed overview of the PER Final releaser's operational capabilities, technological foundations, and data strategies.

4.2.2. Core Technological Components

The following presents a schematic overview of the core technological components of the PER use-case and their respective roles, indicating whether each module is newly introduced or has been updated in the final release:

- **UDT –> Updated** The UDT serves as the central data fusion and sense-making layer for the PER system. It provides a dynamic, semantically enriched virtual representation of the selected area in Venice. By integrating data from several data sources, the UDT models current conditions—such as street adjacency, hazard zones, crowd density, and bandwidth availability. This enables informed scenario planning, situation monitoring, and real-time updates for evacuation strategies.
- **Reinforcement Learning Engine (RL) –> Updated** The RL engine plays a critical role in computing and refining personalized evacuation routes. It consumes real-time input from the UDT and adapts to evolving urban dynamics—such as crowd behavior and emergent hazards. Through continuous learning and simulation feedback, the RL engine produces optimized routing recommendations tailored to individual mobility profiles and environmental conditions.
- **Simulator –> Updated** The Simulator interacts closely with the UDT and RL engine to emulate the movement of individuals and crowds during evacuation scenarios. It supports both training and testing phases by simulating step-by-step behavior based on RL output. This allows validation of strategies in a risk-free environment and contributes to iterative system improvement through feedback loops.
- **Data Mining Framework –> Updated** This framework coordinates the ingestion, processing, and orchestration of data across components. By managing heterogeneous datasets through the DMF and EXTRACT Data Layer, it ensures that all modules receive timely and relevant inputs. It also enables inference-as-a-service via KServe, facilitating scalable deployment on edge devices when needed.
- **Evacuation Mobile App (EMA) -> NEW** The EMA is an Android-based mobile interface designed to assist individuals during emergency situations. It continuously tracks the user's location and, when an emergency is detected, provides real-time, adaptive routing instructions toward the nearest safe zone. The app leverages 5G connectivity and privacy-preserving data handling, enabling fast and secure bidirectional communication with the backend system.

While the UDT, RL engine, Simulator, and DMF form the core infrastructure of the PER use-case, the integration of the EMA and the 5G MEC prototype represent a critical advancement toward operational deployment and citizen engagement.

4.2.3. PER Use-Case Code Repositories

Following, we provide direct hyperlinks to the GitLab repositories containing the source code developed within the scope of this project. Each repository includes documentation and implementation details relevant to the corresponding project components.
https://gitlab.bsc.es/extract/extract-use-cases/per/UDT_v.1

The link above contains the full codebase of the Urban Digital Twin (UDT) developed in the EXTRACT project. It includes all core microservices such as UDTSimulator, DeviceDataConsumer, TrafficData, DataSource2ttl, StateGenerator, CopernicusDataService, and UDT_Dashboard. These components work together to simulate mobile devices, process real-time traffic data, generate semantic urban states, and visualize the city through interactive maps. All services are containerized and designed for deployment in a Kubernetes environment.

https://gitlab.bsc.es/extract/extract-use-cases/per/rl-agent

The gitlab reference above contain the Python scripts for the RL Agent and the connected components. Other the RL training module, in the repository there is the code for the dedicated Movement Simulator that provide assistance to the RL Training module. The two modules are mean to be itegrated in the same environment for a better and faster communication

4.2.4. Urban Digital Twin: Final Implementation and Integration in PER

The Urban Digital Twin (UDT) forms the central integration, fusion, and reasoning layer of the PER system. It semantically models the urban environment by combining static and dynamic data to generate a continuously updated representation of the city's operational state. This section presents:

- the conceptual role of the UDT in the PER use-case;
- its integration with the RL module;
- the semantic model and ontology used;
- the microservices and updated architecture supporting real-time data ingestion and processing;
- the 5G MEC-powered feedback loop for real-time interaction with mobile devices;
- and the role of each key UDT microservice in the final deployment pipeline.

The UDT provides a real-time semantic model of Venice, enabling adaptive evacuation strategies. It supports state extraction for the RL engine and connects to the Simulator and Evacuation Mobile App(EMA) through a 5G-enabled feedback loop. Key innovations introduced in the final version include a refined modular architecture based on independent microservices and an RDF-based semantic model. The integration of real-time sensor data and spatial information leverages a simplified km4City ontology extended with SOSA/SSN for observations, OTN for transport networks, and GeoSPARQL for spatial reasoning. The Virtuoso triplestore stores both static and dynamic graphs, which are periodically queried to construct state snapshots of the urban environment. Each service in the UDT pipeline—ranging from device tracking to traffic aggregation, semantic translation, and state generation—operates

asynchronously via Kafka and is containerized for Kubernetes deployment. The 5G MEC component acts as the interface between mobile devices and the inference engine, mediating bi-directional communication and contextualized instruction delivery. Together, these elements allow the UDT to act as the system's intelligence core, enabling the closed-loop decision-making cycle that defines the PER use-case.

4.2.3.1. Semantic Architecture and Ontology Design

A significant feature within the UDT framework is the amalgamation of dynamic observational data with static urban feature data using a semantic aware approach. This implies the definition of an ontology representing a semantic model of the city managed through the Virtuoso triple store. This integration employs the W3C-OGC SOSA (Sensor, Observation, Sample, and Actuator) ontology, aligning real-time observations with the existing city model to create an up-to-date representation of the urban landscape status. Furthermore, the UDT employs a specialized Python component that interacts with the Virtuoso triple store extracting integrated data to construct a JSON file that delineates the city's status at the time of elaboration. This file subsequently informs the Reinforcement Learning component developed by BSC, facilitating advanced urban dynamics analysis and simulation. Since the extraction of the state of the city is a statical representation that has a time-boxed validity (e.g. after a certain period the distance traveled by a person becomes significant and it is necessary to update the status again), the update frequency will be a parameter that will impact the effectiveness of the service. For the MVP we have not stressed this aspect, preferring to focus on the dataflow and training phase. However, in the final release, this parameter will be carefully assessed in a plausible condition.

RoadGraph Service (Offline Component)

The **RoadGraph service** represents an essential offline tool developed by LRI to generate the static structure of the urban environment, primarily the road network and related geospatial features, used as part of the UDT infrastructure.

4.2.4.1. Overview

The service consists of two core Python scripts designed to process OpenStreetMap (OSM) data into RDF triples aligned with the project's urban ontology, enabling semantic integration with dynamic urban observations.

4.2.4.2. Script Components and Functionality

graphGenerator.py

Extracts geographical and structural road data from OpenStreetMap.

Generates a **GraphML** file encoding the road network structure.

Outputs include nodes, edges, and spatial attributes of roads and intersections.

Visualizes the graph structure (e.g., Figure 11 in D1.2).

osm_road2ttl.py

Transforms the GraphML file into RDF format, aligning with the project's ontology.

Maps nodes and roads to ontology classes such as OTN:Road_Element and GEO:Geometry.

Adds semantic relationships (e.g., starts_at, ends_at, hasGeometry) for spatial reasoning.

Estimates road element widths when OSM data lacks explicit values, based on attributes like lane count and road type.

4.2.4.3. Output and Integration

The RDF file generated from the above scripts encapsulates:

Static roads and intersections

Geometric shapes and coordinates

Topological and semantic links between road segments.

This output is **manually uploaded to the Virtuoso triple store**, stored in a **dedicated static graph**, and is **not updated in real time**, preserving its use for contextualization of live data.

This static graph serves as the **backbone for mobility reasoning**, enabling alignment of real-time device positions and traffic metrics with a pre-defined urban layout.

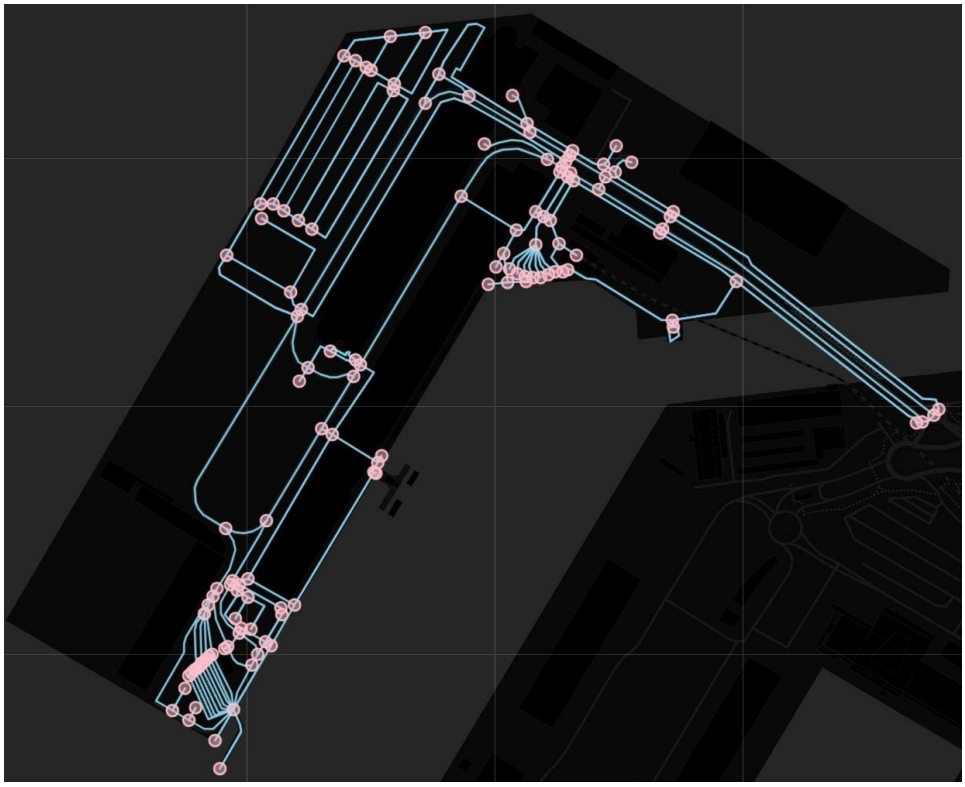

***Figure 7**: Generated Graph*

4.2.4.4. Ontology Description and Semantic Model

The Urban Digital Twin ontology developed for the PER use-case remains based on a modular integration of:

km4City (customized and simplified)

OTN (Open Transport Network) for road graph representation

SOSA for observations

GeoSPARQL for geometry

Schema.org and **Time ontology** for events and temporal relationships

It is structured into **four macro-classes**:

Road (infrastructure)

Observations (real-time and traffic)

Features (physical urban elements)

Events (urban occurrences)

4.2.4.5. Ontology Logic

The semantic structure and class hierarchy described in D1.2 remain valid:

EXT.Road_Element instances represent road segments and are connected via starts_at and ends_at to EXT.Node.

Real-time observations (EXT.DeviceObservation, EXT.TrafficObservation) are subclasses of SOSA.Observation with spatial (GEO.hasGeometry) and temporal (EXT.ObservationTime) annotations.

Urban features like EXT.Building, EXT.GreenField, EXT.Bridge extend GEO.Feature with geometry links.

Events like EXT.Fire, EXT.ChemicalLeak, EXT.Flood are aligned to schema:Event and time:Instant.

Refer to Figures 12–15 in D1.2 for UML diagrams of each class group.

4.2.4.6. New Graph Separation Strategy for RDF Storage

While the **ontology itself remains unchanged**, the **way graphs are stored in the Virtuoso triplestore has been updated** for performance:

Real-time observations (devices, roads, nodes) are now stored in three **dedicated graphs**, e.g.:

http://example.org/graph/realtime/devices_data

http://example.org/graph/realtime/traffic_data

These graphs are **recreated every 30 seconds** with fresh data.

Historical observations are stored in separate daily graphs (e.g., one per date).

This separation avoids querying over large graphs, which previously included both real-time and historical data, significantly improving SPARQL query performance and scalability.

4.2.4.7. UDT Architecture – Updated Version

The updated UDT architecture follows a microservices-based design, replacing the previously monolithic workflow described in D1.2. Each service operates independently and communicates via Apache Kafka. This modular structure supports flexible deployment in environments, enables horizontal scalability, and facilitates continuous integration of new components. All services are containerized using Docker and

deployed via Kubernetes, with each microservice residing in its own folder (see Figure 1). Dedicated deployment scripts and Dockerfiles are provided for each. The architecture introduces a real-time data pipeline, semantic reasoning layer (Virtuoso), and SPARQL-driven state extraction engine, forming the digital backbone of the PER use-case.

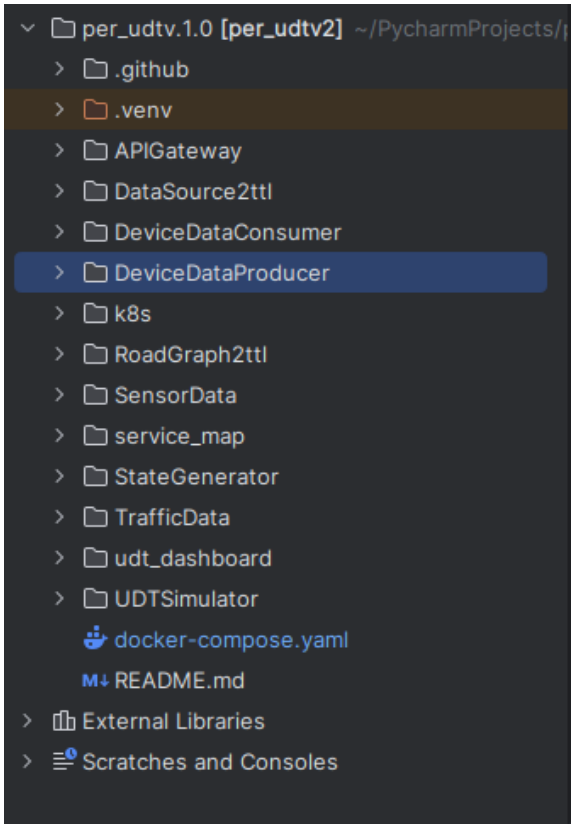

Figure 8 : UDT Microservices Codebase Structure *Folder structure of the PER UDT (implementation, showcasing the modular organization of core microservices, simulation components, and deployment scripts. Each folder corresponds to an independently containerized service supporting real-time data processing, semantic integration, and feedback loops within the PER architecture*.

4.2.4.8. Real-Time Mobility Feedback Loop via 5G MEC

A central innovation of the PER use-case is the **bi-directional, low-latency interaction loop** between mobile devices (real or simulated) and UDT, enabled by **WebSocket-based communication through a 5G MEC (Multi-access Edge Computing)** endpoint.

4.2.4.9. End-to-End Flow Description

A central innovation of the PER use-case is the **bi-directional, low-latency interaction loop** between mobile devices (real or simulated) and UDT, enabled by **WebSocket-based communication through a 5G MEC (Multi-access Edge Computing)** endpoint.

1) **Device Position Updates.** Both real mobile phones (running the EMA application) and simulated devices (from the Device Movement Simulator) send their current positions every 10 seconds. Communication occurs via persistent

WebSocket connections to the 5G MEC service endpoint, which maintains a unique channel per device.

2) **Data Forwarding to UDT.** The 5G MEC does not perform heavy processing but forwards the positional updates as a Kafka stream to a device_topic managed by the UDT's Kafka cluster.
3) **UDT Real-Time State Generation.** The UDT continuously processes the device_topic stream to maintain the latest snapshot of all devices. For each update cycle, the UDT:
   - Keeps the latest position per device (via compacted device_state topic)
   - Groups devices by road segments and intersections
   - Counts the number of devices per road and node
   - Calculates the available bandwidth per road (based on load and density models).
4) **City State Export.** This dynamic city snapshot is transformed into a structured JSON file and pushed to SkyStore every cycle (e.g., every 30 seconds). This output represents the current operational state of the city and is designed for consumption by the RL inference engine.
5) **Action Inference and General Policy Response.** The Inference component, deployed at the edge, continuously processes the latest city state and pre-trained RL policies. The inference output is not device-specific but general: for example, *"in node X with 10 devices, 3 should go left and 7 right."* This general decision avoids coupling actions to specific device IDs.
6) **Device-Specific Action Translation (5G MEC).** The 5G MEC component is responsible for translating general policies into per-device actions. Based on the actual devices currently located in node X, it randomly assigns the instructed movement to specific devices (e.g., Device A → left, Device B → right, ...).
7) **Action Delivery via WebSocket.** The personalized actions are then pushed back to each individual device (phone or simulator) via the same WebSocket channel. Devices react to the instructions, move accordingly, and send their next position after 10 seconds—completing the feedback loop.

4.2.4.10. Microservices and Their Functions

Below is a detailed overview of each component:

1. UDTSimulator. Simulates the movement of mobile devices in the urban area. Posts geospatial updates (position, direction, speed) to the device_topic via REST API.

Attribute	Details
Function	Simulates mobile agents moving in the city and manages bidirectional WebSocket communication with the 5G MEC.
Technology Stack	Python, WebSocket (client), asyncio, Docker, Kubernetes
Internal Logic	For each simulated device, opens a WebSocket connection with the 5G MEC. Sends updates (position, speed, direction) every 10 seconds. Listens for device-specific instructions returned by the MEC.
Deployment	Standalone pod running multiple async agents inside UDT Kubernetes namespace.
Main Output	WebSocket messages with geospatial data sent to the 5G MEC.

Used By	5G MEC Component
Benefits	Allows large-scale testing of the UDT pipeline with real-time emulation of mobile devices and bidirectional interaction.

2. 5G MEC Component – Edge Gateway for Device Interaction. The 5G MEC (Multi-Access Edge Computing) service represents the updated evolution of the device_producer component. It is now implemented as a standalone microservice deployed at the edge level of the UDT pipeline and is responsible for mediating real-time communication between simulated/real mobile devices and the core Kafka-based data infrastructure of the UDT system.

Key Responsibilities	• **Real-time Position Updates**: Maintains persistent WebSocket connections with mobile devices (EMA app or simulators). Devices send updated position, speed, and direction every 10 seconds.• **Secure Kafka Forwarding**: Sends received device updates to the UDT Kafka cluster via a **Kafka gRPC Proxy**, avoiding direct Kafka exposure.• **Inference Querying**: Periodically queries the RL inference module to retrieve group-based movement instructions (e.g., "group 3 left").• **Per-Device Action Assignment**: Translates group instructions into individual movement commands based on real-time knowledge of each device's location.• **Bidirectional Feedback**: Sends individualized instructions back to each device via WebSocket, enabling closed-loop interaction.
Internal Logic	For each connected device, the MEC service handles a persistent WebSocket session. It forwards each incoming update to a Kafka topic using a secure, authenticated gRPC proxy connection. At regular intervals, it queries the output of the RL module to retrieve general movement plans. These plans are interpreted in the context of current device states to determine appropriate per-device actions, which are then dispatched back to each agent.
Technological Overview	• **Programming Stack**: Python with asyncio and grpc.aio for scalable, asynchronous operations.• **Kafka Communication**: Kafka gRPC Proxy deployed in Kubernetes, using TLS and token-based authentication.• **WebSocket Integration**: Full-duplex channel enabling real-time updates and responses per device.• **Security**: Mutual TLS authentication, token-based access control for Kafka publishing.
Used By	• **DeviceDataConsumer**: Subscribes to device_topic to receive updates forwarded by the MEC.• **Simulated or real mobile devices**: Send data and receive movement instructions.• **RL inference module**: Provides input for group-based movement planning.
Benefits	• Significantly **reduces latency** by executing logic at the edge of the system.• **Decouples Kafka** from external exposure through proxy-based architecture.• **Enables real-time integration** between AI-driven strategies and mobile agent behavior.• **Scales securely** to thousands of concurrent device connections using event-driven design.

Figure 9 : FastAPI Interface for UDT Command and Control API *Snapshot of the interactive FastAPI documentation for the PER UDT system, illustrating available REST endpoints for retrieving device, road, and node data, checking system health, and issuing broadcast or unicast evacuation instructions. This interface supports both monitoring and real-time command dissemination in emergency scenarios.*

3. DeviceDataConsumer Microservice

The **DeviceDataConsumer** service is responsible for processing mobile device position data arriving from the device_topic Kafka stream. It acts as the first semantic filter in the real-time UDT pipeline, continuously maintaining a live snapshot of all known mobile devices and their latest positions. It ensures deduplication by retaining only the most recent message per device ID and publishing it to the compacted topic device_state, which is then used by downstream services such as TrafficData.

Internal Logic	1. **Subscription**: Subscribes to device_topic and receives positional updates from real or simulated devices.2. **Processing**: Parses messages to extract device_id, timestamp, latitude, longitude, speed, and direction. Enriches data with computed attributes such as road_segment or node_id (optional).3. **State Update**: Maintains a **Faust Table** keyed by device_id, storing only the latest known state. Discards older or duplicate updates.4. **Publishing**: Pushes the cleaned, deduplicated stream to a **Kafka compacted topic device_state**, which keeps only the most recent message per key.
Technological Overview	• **Faust**: Stream processing in Python using agents and RocksDB-backed state tables.• **Kafka**: Input (device_topic) and output (device_state, compacted).• **Docker & Kubernetes**: Deployed as a horizontally scalable pod; Kafka partition-based parallelism is supported.• **State Store**: RocksDB-based local state table for real-time lookups and deduplication.
Used By	• **TrafficData**: Uses the device_state stream to calculate road congestion and node density.• **DataSource2ttl**: Transforms the

Benefits	device states into RDF for semantic reasoning.• **Virtuoso (via downstream)**: For storing current device positions. • **Low latency**: Faust provides high-throughput stream processing with microsecond-level event handling.• **Reduced duplication**: Deduplication is built-in using Faust tables and Kafka compaction.• **Scalability**: Scales linearly with Kafka partitions and Kubernetes pod replicas.• **Efficient filtering**: Only the most recent state per device is passed to downstream services, minimizing processing overhead.
Output	• **Kafka topic device_state** (compacted): Real-time, deduplicated snapshots of all devices, updated continuously.

4. TrafficData Microservice

The **TrafficData** microservice plays a crucial role in interpreting the mobility patterns of mobile devices within the UDT. By subscribing to the device_state Kafka topic—where each message represents the latest known state of a mobile device—it aggregates this information to compute real-time traffic conditions across road segments and intersection nodes. This service transforms individual device positions into actionable summaries about road congestion and node-level density, publishing the results to two compacted Kafka topics: road_state and node_state.

Internal Logic	1. **Subscription**: Consumes messages from the device_state Kafka topic, receiving the current state of each device.2. **Spatial Grouping**: • Roads are identified by (node1, node2) pairs. Inverse directions (e.g., A → B and B → A) are normalized to represent the same road. • Nodes represent intersections and are detected when node1 == node2.3. **Aggregation**: • **Road segments**: Calculates average speed and device count. • **Nodes**: Counts the number of devices present.4. **Publishing**: Aggregated metrics are continuously pushed to compacted Kafka topics: road_state and node_state.
Technological Overview	• **Faust**: Used for real-time grouping and stateful streaming.• **Kafka**: Subscribes to device_state, produces road_state and node_state (both compacted).• **Docker & Kubernetes**: Runs as a scalable pod in the UDT cluster with horizontal partition-aware scalability.• **State Tables**: Maintains two Faust tables: road_state_table and node_state_table.
Used By	• **DataSource2ttl**: Converts traffic metrics into RDF and stores them in Virtuoso.• **StateGenerator**: Combines traffic data with bandwidth, device state, and events to build the urban state snapshot.
Benefits	• **Real-time traffic intelligence**: Provides constantly updated insights on road congestion and node density.• **Efficient computation**: Operates only on the latest known position of each device thanks to device_state compaction.• **Flexible extension**: Can be extended with more metrics like average direction, node pressure, or inferred bandwidth.
Output	• **road_state** (compacted): Traffic metrics per road segment (avg speed, device count).• **node_state** (compacted): Device counts per intersection node.

5. DataSource2ttl Microservice

The **DataSource2ttl** microservice is responsible for mapping real-time urban data into semantically structured RDF triples using a domain-specific ontology. It acts as the main interface between streaming mobility data and the RDF knowledge base stored in Virtuoso. This service ingests data from Kafka topics (device_state, road_state, and node_state) and updates the semantic layer with spatial and temporal observations for use in reasoning, querying, and downstream applications.

Internal Logic	1. **Kafka Consumption**: Listens to three compacted Kafka topics: • device_state → device positions and speeds • road_state → traffic metrics per road • node_state → node congestion and density2. **Ontology Mapping**: Maps input data into RDF triples using SOSA/SSN, OTN, and GeoSPARQL: • DeviceObservation: device location, speed, timestamp • TrafficObservation: per-road and per-node metrics3. **Graph Management**: • **Real-time graph**: Rebuilt every 10 seconds (e.g., graph:realtime) • **Historical graph**: Appends data every 15 minutes using timestamped named graphs (e.g., graph:history_YYYYMMDD_HHMM)4. **Virtuoso Update**: Sends SPARQL Update requests via HTTP to delete stale triples from the real-time graph and insert new ones. Historical triples are appended without deletion.
Technological Overview	• **Python 3.10+**: Core implementation• **Kafka**: Inputs from device_state, road_state, and node_state• **RDFlib**: In-memory RDF graph handling and TTL serialization• **Virtuoso**: SPARQL endpoint used for semantic storage and updates• **Ontology Stack**: SOSA/SSN (sensor observations), OTN (road network), GeoSPARQL (geospatial linking)• **Kubernetes**: Runs in UDT namespace as an independent microservice with both streaming and scheduled logic
Used By	• **StateGenerator**: Queries graph:realtime to build the semantic city state snapshot.• **Virtuoso consumers**: Any component accessing the SPARQL endpoint for semantic queries.
Benefits	• **Semantic enrichment**: Enables reasoning and querying over raw mobility and traffic data.• **Efficient querying**: Real-time graph always contains the latest data only.• **Temporal traceability**: Historical graphs preserve evolution of the system for time-series or offline analysis.
Output	• Virtuoso named graphs: • graph:realtime – updated every 10s • graph:history_YYYYMMDD_HHMM – appended every 15 minutes

6. StateGenerator Microservice

The **StateGenerator** microservice serves as a core component in the UDT system for periodically extracting the current semantic representation of the city. It transforms RDF data from the Virtuoso triplestore into lightweight and structured JSON files, which are consumed by downstream components such as the RL module, dashboards, and simulation engines.

Internal Logic	1. **Scheduled SPARQL Execution**: Every 30 seconds, the service sends multiple SPARQL SELECT queries to Virtuoso targeting the real-time graph (graph:realtime). It extracts: • Mobile device positions • Road and node congestion metrics • Active event areas with danger/safe points • Bandwidth availability (if available)2. **SPARQL Parsing**: The results of the queries are parsed and organized into a consistent JSON schema grouped by: • Roads • Nodes • Mobile Devices • Event Context3. **File Generation**: Outputs a single .json file that captures a full semantic snapshot of the city.4. **Distribution**: Uploads the file to **SkyStore** (S3-compatible). Sends a notification event so that **RL agents and visual systems** are alerted and can react to the new state.
Technological Overview	• **Python 3.10+**: Used for querying, parsing, and scheduling.• **SPARQL**: Standard query language for RDF used to extract information.• **Virtuoso**: Accessed via HTTP endpoint.• **SkyStore (S3-compatible)**: Used for storing .json snapshots and enabling external access.• **Scheduler**: Executes the query process every 30 seconds.
Used By	• **RL Module**: Uses the state file to determine next movement instructions.• **5G MEC**: Fetches device-level actions from the updated file.• **UDT Dashboard / Simulator**: Visualizes the city's current state using the JSON file.
Benefits	• **Real-time semantics**: Extracts and transforms RDF data into immediately usable form.• **Efficient integration**: JSON output is lightweight and fast to process by AI modules.• **Modular and extensible**: Additional SPARQL modules (e.g., environment data) can be plugged in.
Output	• JSON file (city_state_YYYYMMDD_HHMMSS.json) uploaded every 30 seconds to SkyStore.• Event notification to downstream consumers (RL, 5G MEC, Dashboard).

Figure 10 : *Example of JSON data structures used in the PER use-case for tracking individual positions and managing dynamic event information. The data supports real-time decision-making in the evacuation routing system.*

7. UDT_Dashboard Microservice

The **UDT_Dashboard** microservice provides a real-time visual interface for monitoring the status of the UDT. It serves both as a control panel and a situational awareness tool for project stakeholders. The dashboard is designed to render live data—such as mobile device positions, event areas, and recommended evacuation paths—through intuitive 2D and 3D map views.

Internal Logic	1. **Real-Time Tracking**: Connects to REST endpoints exposed by the APIGateway to retrieve: • Live positions of mobile agents • Congestion and traffic state summaries • Active events and designated safe/danger points2. **Visualization Layer**: • **CesiumJS** renders 3D urban views with terrain, buildings, and dynamic mobile agents. • **Leaflet.js** renders 2D overlays for mobile density, heatmaps, zones, and event boundaries.3. **Device State Rendering**: Devices are shown using color-coded indicators based on their status (e.g., idle, evacuating, unsafe).
Technological Overview	• **CesiumJS**: 3D geospatial viewer for terrain, buildings, and entities.• **Leaflet.js**: Lightweight 2D map engine for overlays and markers.• **JavaScript + HTML/CSS**: Standard frontend interface and UI rendering.• **RESTful APIs**: Consumes UDT backend endpoints for live data.• **HTTPS Web App**: Served as a browser-accessible user interface.

Used By	• **Project stakeholders and operators**: For real-time monitoring, scenario tracking, and manual validation.• **Simulation testers**: To visually verify mobile movement patterns, safe zones, and RL-driven outcomes.
Benefits	• **Situational awareness**: Offers an intuitive, live visualization of all UDT components and dynamics.• **User-friendly**: Lightweight and accessible from any modern browser without additional installation.• **Insightful UI**: Clearly shows risk zones, evacuation flows, and mobility statistics at a glance.
Output	• **Web-based dashboard**: Displays interactive 2D/3D maps with real-time overlays of devices, traffic, and events.

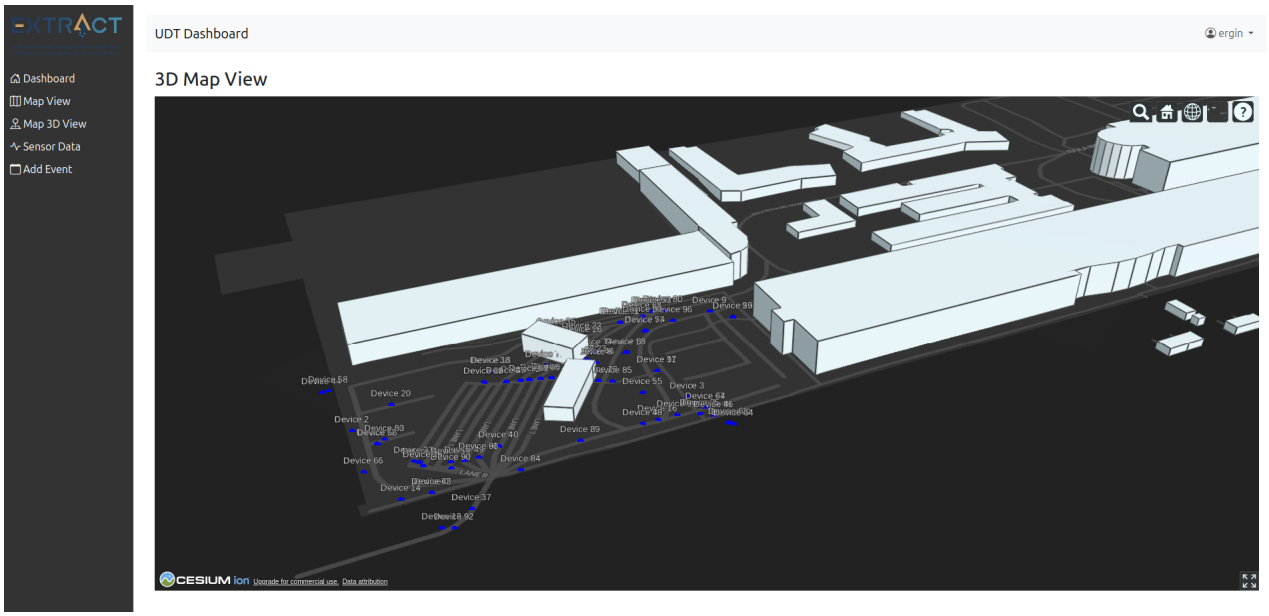

Figure 11 : *User interface of the EXTRACT UDT Dashboard for adding and managing danger events. The dashboard allows manual input of event parameters, such as type and location, and associates danger and safe nodes with each event for real-time evacuation management*

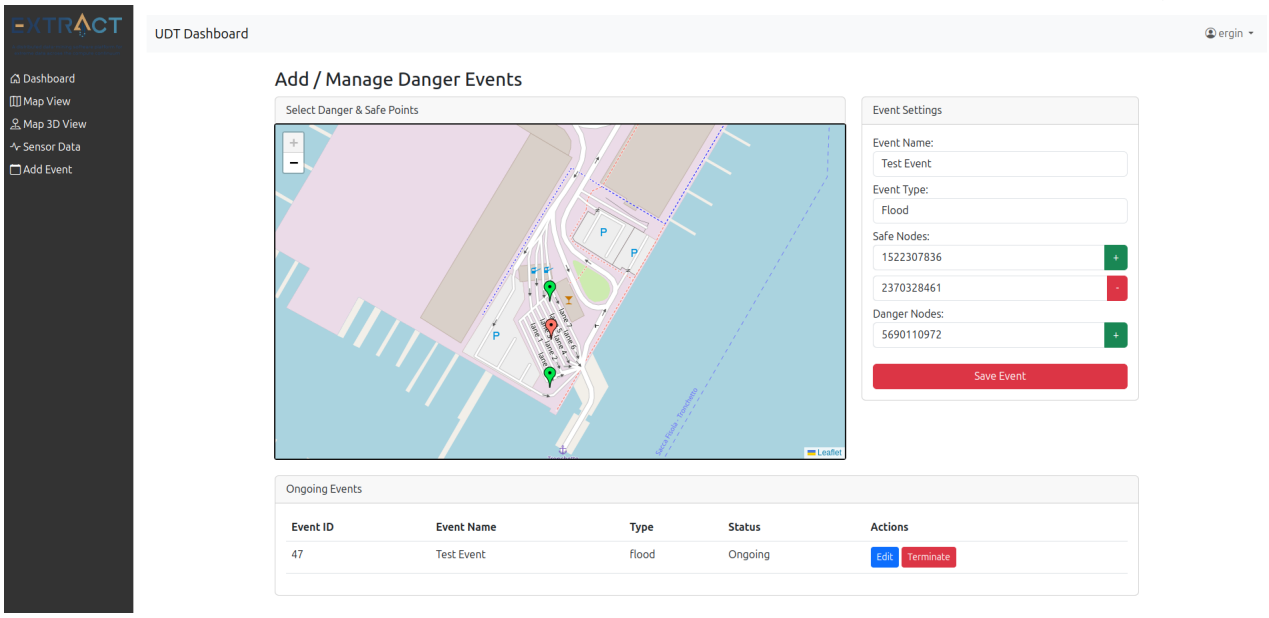

Figure 12 : *3D Map View of the EXTRACT Urban Digital Twin (UDT) Dashboard, visualising the spatial context of monitored environments. This interface supports situational awareness by displaying sensor data and event-related information within a three-dimensional urban model.*

To ensure our system's state representation is both accurate and dynamic, we employ SPARQL queries to interact with a Virtuoso endpoint, fetching dynamic data about the urban digital state. This process is fundamental to our approach, allowing us to retrieve up-to-date information on traffic movements and resource utilization across the network. Upon performing these queries, the retrieved data is systematically processed and stored into our designated tensors. Utilizing the SPARQL protocol, we execute queries against the Virtuoso endpoint, a powerful database engine optimized for storing and querying RDF (Resource Description Framework) data. These queries are designed to extract the latest information regarding the number of individuals at nodes (intersections or points of interest) and on edges (paths or roads connecting these nodes), as well as the current status of bandwidth or other resources on the network. By integrating the SPARQL query results into our tensor-based state representation, we achieve a dynamic and accurate reflection of the urban digital state The output is collected by the RL agent to base its decisions on the most current data, optimizing traffic flow and resource distribution with a high degree of precision and effectiveness.

8. CopernicusDataService

The CopernucusDataService is a backend microservice designed to retrieve and process geospatial and environmental datasets from the **Copernicus Emergency Management Service (EMS)** and related sources (e.g., GloFAS and Sentinel Hub). Its role is to enrich the Urban Digital Twin (UDT) with authoritative external data, particularly hydrological forecasts and satellite imagery, supporting emergency preparedness and urban resilience in the PER use-case. This service operates in coordination with the semantic layer, triggering updates to the Virtuoso triplestore when relevant indicators or thresholds are met. By integrating near real-time flood

risk and remote sensing observations, it contributes to decision-making components such as the Reinforcement Learning agent and alert generation subsystems.

Internal Logic	1. **Input Retrieval**: • Queries the Copernicus Climate Data Store (CDS) via cdsapi for GloFAS flood forecasts. • Accesses Copernicus Marine Service via motuclient for ocean data (e.g., sea level). • Downloads satellite imagery using the SentinelHub API.2. **Processing Pipeline**: • Filters data based on geographic bounding box and time. • Extracts relevant geospatial layers (e.g., rainfall intensity, flood polygons). • Formats results as structured JSON.3. **Output Handling**: • Forwards results to DataSource2ttl for semantic mapping. • Optionally produces events to Kafka.4. **Coordination with Virtuoso**: When new indicators exceed thresholds (e.g., rising water levels), triggers RDF updates in Virtuoso.
Technological Overview	• **Python** (async-based implementation for concurrency)• **cdsapi**: Access GloFAS and CDS datasets• **motuclient**: Retrieve marine datasets (e.g., sea conditions)• **Sentinel Hub SDK**: Access and process satellite imagery• **Optional Kafka integration**: For pushing alerts• **Virtuoso integration**: Via DataSource2ttl
Used By	• **DataSource2ttl**: Receives pre-processed environmental data to create RDF observations.• **StateGenerator and RL**: Use updated Virtuoso graphs for generating state files and adapting AI recommendations.• **Alert systems**: May use the JSON results to notify or trigger warnings.
Benefits	• **Enhances urban resilience**: Brings authoritative environmental intelligence into the UDT.• **Supports real-time decision making**: Enables proactive responses based on external flood/sea/imagery data.• **Seamless integration**: Works alongside semantic and reasoning layers.• **Extensible**: New layers, formats, and thresholds can be easily added.
Output	• **Structured JSON files** containing Copernicus data (e.g., flood extent, sea level)• **Semantic RDF** observations forwarded to Virtuoso via DataSource2ttl• **Time-stamped updates** for both real-time and historical graphs
Deployment	• Deployed as a standalone service in the UDT Kubernetes cluster• Requires access tokens or credentials for CDS, SentinelHub, and Marine services

4.2.4.11. Emergency Notification and Urban Digital Twin Update

Notification of Emergency Events

The initiation of the PER Use-Case Pilot's operational cycle begins with the receipt of an emergency notification, a critical component that underpins the swift and effective response to incidents within Venice. This process is triggered by an alert from Venice's integrated emergency management system, which signals the presence of a potential threat, such as a fire in a specific area of the city.

This alert activates a series of protocols designed to update the UDT with the latest situational data, ensuring that the response is informed by the most current and comprehensive understanding of the city's status.

Updating the Urban Digital Twin

Following the emergency notification, the UDT undergoes a crucial update process to integrate diverse datasets that collectively offer a detailed and nuanced view of Venice's current state. This update includes data from the SCR, which aggregates key information relevant to emergency management, such as infrastructure status, population distribution, and environmental conditions. Additionally, the precise locations of mobile devices are refreshed using the High Accuracy Positioning (HAAP) system, incorporating real-time positional data that is critical for managing evacuation and response efforts. The Copernicus system contributes satellite-derived information, enriching the UDT with broader contextual data. However, it is worth to remark that this part will not included in the MPV. This phase also involves the invocation of reinforcement learning mechanisms to update the location of individuals within the simulation environment, initiating a training phase tailored to the emergent scenario reflected in the UDT's new state. This process ensures that the deployment of RL models to edge devices and the subsequent instruction dissemination to mobile phones in the affected area are based on the latest, most accurate city model. As new RL models are trained and deployed, the cycle of updating the UDT and refining evacuation strategies continues, with real-time feedback from mobile phones constantly informing the UDT.

4.2.5. Reinforcement Learning for Evacuation Optimization

As part of the EXTRACT orchestration logic applied to the PER use-case, a critical advancement since the previous phase involves the consolidation and refinement of the Reinforcement Learning (RL) framework that governs the evacuation routing logic. This component is triggered immediately following an emergency event notification and the corresponding update of the Urban Digital Twin, and it serves as the core mechanism for dynamically computing evacuation strategies under uncertain and evolving conditions.

The complexity of Venice’s urban layout—with its dense pedestrian network, constrained bridges, and highly dynamic risk zones—renders classical, rule-based routing approaches inefficient or brittle in real emergencies. Reinforcement Learning was chosen from the very beginning of the project as a paradigm not only for its ability to learn adaptive policies through interaction, but also because of its flexibility in handling multi-agent coordination, variable constraints, and non-linear feedback.

Unlike static routing algorithms, the RL agent learns evacuation strategies by simulating a multitude of possible scenarios. Through this process, it is able to uncover optimal or near-optimal policies that minimize evacuation time and congestion, while balancing safety, route length, and adaptive responses to unexpected events (e.g. street flooding, crowding, blocked access points).

In the current implementation, the **RL model has evolved beyond a proof-of-concept** into a more robust, modular system with improved simulation accuracy, training convergence, and real-time inference capabilities, tested in larger maps with good results.

RL Algorithm: Branching Dueling Q-Network

The reinforcement learning algorithm finally employed in the PER use-case is Branching Dueling Q-Network (BDQ). This choice is motivated by the algorithm's capability to handle high-dimensional discrete action spaces, enabling the creation of actions that represent the simultaneous movement of multiple entities, such as multiple persons.

Alternative algorithms like DQN were dismissed due to their incapacity to manage high-dimensional action spaces; essentially, they could only handle the movement of a single individual at a time. Conversely, Branching Dueling Q-Network (BDQ) has demonstrated efficacy across diverse tests encompassing varying map sizes and population densities.

Continuous action space algorithms like DDPG and MARL have been discarded in this project. DDPG was unable to learn the policy effectively, yielding suboptimal results across tests. While MARL has theoretical advantages in modelling interactions between individuals, its implementation is highly complex and beyond the scope of this project. Instead, BDQ has been adopted due to its superior performance and its ability to handle high-dimensional discrete action spaces efficiently.

The following two subsections (State and action representation, and Reward), are some explanations that were already presented in D1.2, but which we have decided to keep with minor additions due to its very technical ground and by being very relevant to understand the way the RL operates.

State and action representation

The data extracted from OSM is structured as a graph, wherein streets are depicted as edges and intersections as nodes. Alongside this representation, OSM provides valuable details such as the distances associated with each edge. Furthermore, our analysis incorporates additional factors including population density at nodes and edges, as well as the bandwidth of each street, which denotes the capacity for pedestrian movement.

The data is structured within an array featuring two channels: bandwidth (BW) and the number of individuals. The distances for each street are stored in a separate array, although these distances remain static and are not utilized to depict the present state. In our approach to optimize memory allocation and computation times, we focus exclusively on the dynamic data of BW and people.

The BW channel effectively communicates the capacity of a street, indicating the number of individuals it can accommodate at any given moment. For instance, if a street initially had a capacity of 100 persons and 20 people are currently traversing it, the BW value would be adjusted to 80. It also serves as a representation of paths that are no longer accessible, achieved by setting the BW to 0.

The people channel denotes the current population within a specific street or intersection, providing a real-time data of human presence.

Initially, we employed a torch tensor adjacency matrix to convey this information. However, this approach proved computationally expensive. Consequently, we transitioned to an adjacency list—a 1D array containing all pertinent details about nodes and edges. Each piece of information is stored in a separate channel, resulting in a structure with dimensions ((n_nodes + m_edges), 2), where 2 represents bandwidth and persons. The order follows a pattern: information for all nodes precedes that for all edges. Furthermore, the data is organized based on Open Street Map IDs.

To facilitate the transition, we have implemented an index mapping array that precisely denotes the actual positions within the adjacency array. This mapping array mirrors the length of the state (comprising n_nodes + m_edges) and aligns each position with the corresponding node or edge within the state array. For instance, the value at index 0 of the state array of a graph with 10 nodes and 18 edges corresponds to the value (0, 0) in the mapping array, signifying node 0. Similarly, at index 11 of the mapping array, a value such as (0, 4) indicates that position 11 in the state array holds the value of the edge connecting node 0 to node 4.

Below is an illustrative example demonstrating the functionality of a RL agent navigating a simplistic map consisting of merely 10 nodes. It's important to note that the data presented here is hypothetical and intentionally simplified for clarity in showcasing the concept.

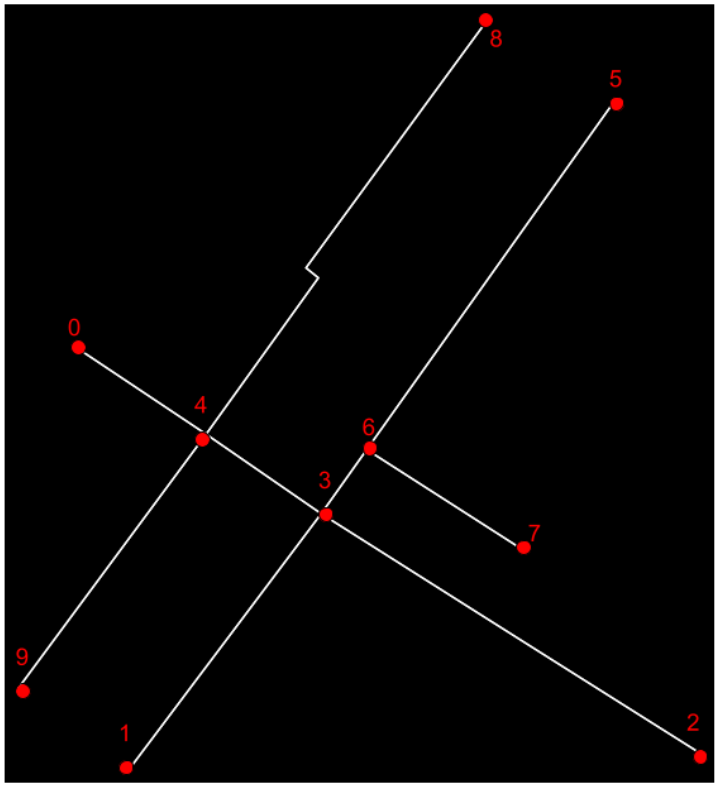

Figure 13 : *Graph-based representation of the simulated environment used in the PER use-case. Nodes represent key locations, and edges define possible pedestrian movement paths used for evacuation route planning and simulation*

The following array represents the amount of people found in the map:

[**4**, **7**, 0, 0, 0, 6, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

So there are 4 persons in the node 0, 7 persons in the node 1, etc. So if a person moved from node 0 to 4, then the value found in index 11 in the state would represent that movement.

The action representation has been modified for the BDQ algorithm, as it does not support continuous action spaces. Instead, the agent considers a discrete set of possible distributions of people across nodes and selects the best action accordingly. For example, if a node has two outgoing edges, the agent chooses from predefined

discrete distributions such as [(0, 1), (0.1, 0.9), (0.2, 0.8), ..., (1, 0)], representing the proportion of people sent along each edge. This discretization enables BDQ to efficiently handle multi-dimensional action spaces by branching over each action dimension.

Nodes 0 and 1 each have only one outgoing connection, so their action space consists of a single possibility: (1), meaning all individuals are sent through the only available edge. This information is stored in a dictionary where the keys represent the node IDs, and the values indicate the index of the selected action. For example:

{0: 0, 1: 0}

In this case, the value 0 signifies that there is only one available action (i.e., one possible distribution), so no choice is needed—the agent always selects that sole option.The array below illustrates the available bandwidth for each street, this value is not considered in intersections:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, **4**, **6**, 6, 6, 6, 6, 6, 5, 6, 6, 5, 5, 6, 5, 6, 6, 6, 5]The conversion array (numpy) that represent the corresponding nodes and edges:

[**(0, 0)**, **(1, 1)**, (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), **(0, 4)**, **(1, 3)**, (2, 3), (3, 1), (3, 2), (3, 4), (3, 6), (4, 0), (4, 3), (4, 8), (4, 9), (5, 6), (6, 3), (6, 5), (6, 7), (7, 6), (8, 4), (9, 4)]

The previously described action forces the maximum number of people to be sent through the outgoing edges, constrained by bandwidth limitations:[**0**, **1**, 0, 0, 0, 6, 0, 0, 0, 5, **4**, **6**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

When these persons reach their destination node the state will look like the this:

[**0**, **1**, 0, **6**, **4**, 6, 0, 0, 0, 5, **0**, **0**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Reward

Reward is a fundamental concept within reinforcement learning algorithms. It is a scalar value provided by the environment to the agent at each time step, indicating how well the agent is performing. It serves as feedback to guide the agent towards achieving its objectives.

The primary goal of the agent is to learn a policy (actor) that maps observations to actions in a way that maximizes the expected cumulative reward over time. The critic, on the other hand, evaluates the actions taken by the actor by estimating the expected cumulative reward associated with a particular state-action pair.

Rewards play a crucial role in the exploration-exploitation trade-off. The agent explores different actions to discover potentially better strategies while exploiting known strategies that have led to high rewards in the past.

Various rewards have been encoded and require testing to determine the optimal choice. After comparing the results, the reward that provided best results and was not computationally expensive and simple was:

Count of individuals located within the simulationReturn -1 when everyone still hasn’t been saved. Additionally, 10 points are deducted each time the agent sends a person to a dangerous node. Conversely, 10 points are awarded when every individual in the simulation reaches a safe destination.

Baseline

The baseline is defined by a heuristic algorithm that prioritizes moving the maximum number of people along the fastest available routes to a safe zone. This approach serves as a rigid, rule-based benchmark against which the performance of the RL model is evaluated. The issue with this baseline approach is that it doesn't account for the dynamic conditions of an emergency situation or the varying bandwidth of each street. As a result, choosing the shortest path to safety may not yield the fastest or most effective route, since some streets can become unusable due to these constraints.

This algorithm also supports the agent's learning process. As training progresses, the agent occasionally selects actions based on this baseline, with a probability that decreases over time. This approach helps the model avoid relying solely on random actions and encourages it to favor decisions aligned with the shortest path to a safe zone.

Neural Network architecture

The decision-making system relies on a Branching Dueling Q-Network (BDQ), a variant of Q-learning tailored to environments with multiple discrete actions per decision point. Rather than maintaining separate actor and critic networks, this model combines value-based learning with a branching advantage structure.

Input:The input to the network is the flattened state representation, where the state size is a one-dimensional vector encoding information such as bandwidth, people, or other environment features across nodes and edges.

Shared Hidden Layers:The BDQNetwork begins with a set of shared fully connected layers:

- The first layer transforms the input state vector (state_size) into a hidden representation (hidden_size).
- Subsequent layers maintain the same hidden size, typically ranging from 2 to 4 layers.
- ReLU activation is applied after each layer.

Value Stream: After the shared layers, the value stream outputs a single scalar V, representing the state value, i.e., the expected return from that state regardless of action.

Advantage Streams (Branching): For each branching node, that is, a decision point with multiple action candidates, the model has a dedicated advantage stream. These streams output advantage values A for each possible action at that node.

Combining Value and Advantage: The Q-value for each action at each node is computed using the dueling architecture formula:

$$Q(s, a) = V(s) + A(s, a) - \text{mean}(A(s, \cdot))$$

This ensures that the model learns both how valuable a state is and which actions are relatively better at each node.

Scalability: As the state space (e.g., number of nodes and edges) increases, the input dimension and number of branches grow. Consequently, a larger hidden_size (commonly between 64 and 256) may be necessary to maintain expressive power and performance.

Deployment and Iteration of RL Models

The deployment of RL models to edge devices is a crucial action that enables real-time communication of evacuation instructions to individuals' mobile devices within the affected area. This deployment is not a one-time action but rather part of an iterative process where RL models are continuously refined based on the flow of new data and the performance of previous evacuation instructions. As the Urban Digital Twin receives updates about the city's state and the locations of individuals, these insights are fed back into the RL system to refine and adapt its models. This iterative loop ensures that the RL system remains responsive to the dynamic nature of emergency situations, optimizing evacuation routes as conditions change.

By harnessing the capabilities of RL, the PER Use-Case aims to deliver a highly adaptive and responsive system that can manage the complexities of emergency evacuation in the unique urban environment of Venice.

Results

The RL model provides promising results, by improving on the baseline previously described and almost reaching the theoretical estimation of the best results.

The following diagram shows a comparison of these three approaches on 18 random maps with 50 nodes and 100 people each:

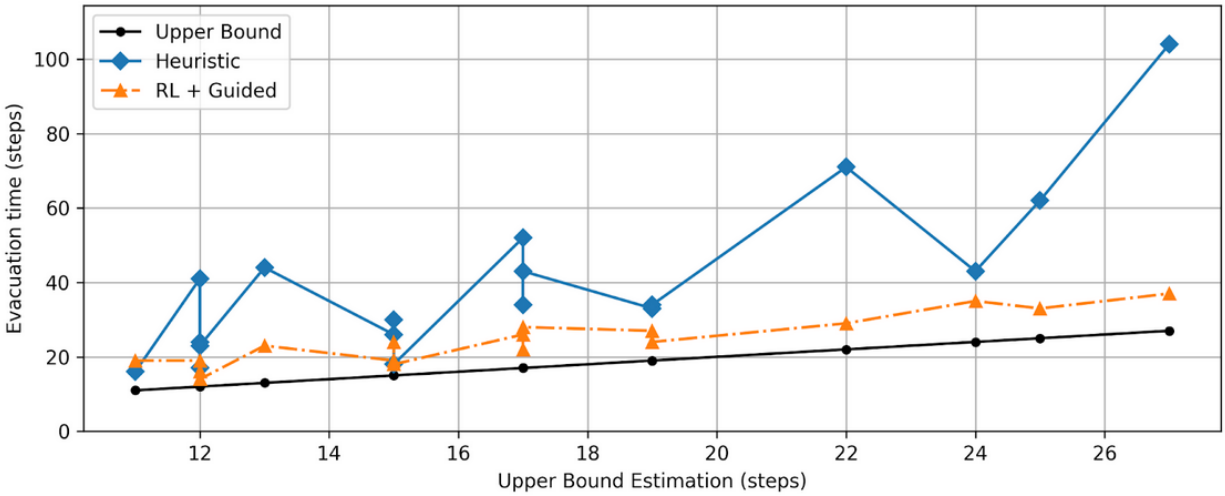

Figure 14 : *Comparison of evacuation performance (in steps) using three different strategies—Upper Bound, Heuristic, and RL + Guided—across varying levels of upper bound estimations. The RL + Guided approach consistently outperforms the heuristic baseline.*

The plot shows evacuation times in steps (one step equals 1 second) on the vertical axis versus the planner's own "upper-bound" estimate of how many steps should be needed (on the horizontal axis). Three curves are shown:

1. **Upper Bound (black circles)**: This is essentially the planner's idealized estimate of the minimum number of steps needed, calculated using binary search. It grows smoothly and linearly from about 10 steps (when the problem is easiest) up to about 27 steps (when it's hardest).
2. **Heuristic Baseline (blue diamonds)**: The pure heuristic strategy often deviates wildly from the ideal.
   - At moderate difficulty (upper bound ≈ 18), the heuristic sometimes takes over 50 steps, more than twice the theoretical minimum.

- In the hardest settings (upper bound ≈ 27) it can even exceed 100 steps, nearly four times the lower-bound estimate.
- Overall, its performance is highly variable and degrades quickly as the problem gets harder.

3. **RL + Guided (orange triangles)**: By contrast, the reinforcement-learning agent, guided by the baseline's heuristic, stays very close to optimal:

- Across all difficulty levels, its evacuation times hover only about 5–10 steps above the theoretical lower bound.
- Even in the hardest scenarios (upper bound ≈ 27), it completes in roughly 35–37 steps, just a small constant overhead compared to the ideal.

This next diagram compares the same results but with 18 maps with 80 nodes and 100 people:

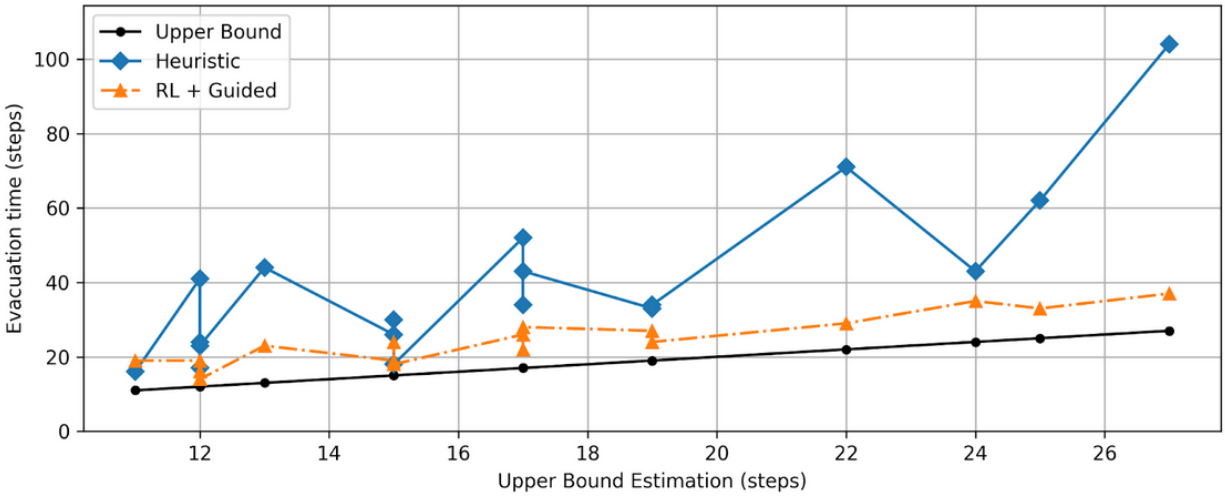

Figure 15. *Overlay of performance results confirming the stability and efficiency of the RL + Guided strategy over multiple simulation runs. Results show lower and more consistent evacuation times compared to the heuristic method as complexity increases.*

1. **Upper Bound**: Continues to grow steadily, as expected.
2. **Heuristic Baseline** : Still shows significant volatility, particularly at mid-range difficulties (upper bound ≈ 13–16), where it can spike up to ~43 steps, despite the theoretical minimum being under 20. While slightly more stable in the lower bound range, it again performs poorly at the higher end (up to 25).
3. **RL + Guided**: While no longer as tight to the optimal as before, still consistently outperforms the heuristic baseline. It maintains a relatively controlled overhead, generally staying within 5–15 steps of the upper bound.

Across both the 50-node and 80-node maps, the RL-guided approach consistently outperforms the pure heuristic, offering significant advantages in both efficiency and reliability. On the smaller map, the heuristic often requires tens of extra steps and becomes increasingly erratic as difficulty grows, while the RL + Guided method stays consistently close to the planner's estimated lower bound, scaling gracefully with problem complexity.

On the larger 80-node map, although the performance gap narrows slightly, the RL model still maintains a clear edge. It avoids the sharp inefficiencies and variability observed in the heuristic, particularly in higher-difficulty scenarios. Overall, the

RL + Guided strategy proves to be more robust and scalable, especially as the environment becomes more complex.

4.2.6. People Movement Simulator

As described in deliverable D1.2 , the People movement simulator is directly used by the RL component responsible to find and elaborate the evacuation routes.

Compared to the previous version, while maintaining its functions unchanged, some components have been refined, trying to optimize the execution of the script and standardize performance and results, in order to allow:

- an optimized execution of the process and capable of satisfying the needs of the platform
- a fluid and streamlined exchange of information, perfecting both the format and the method of data exchange

Motion Simulator for RLTraining

To enable the training of the RL-based evacuation routing system, a custom motion simulator has been developed. Since real-world trial-and-error learning is infeasible in emergency scenarios—especially within the urban complexity of Venice—the simulator provides a safe, controlled, and repeatable environment for testing and validating RL-generated actions. The simulator receives input data from the UDT and applies the actions produced by RL, replicating realistic user movements across the city. It supports a wide range of scenarios, including those that cannot be reproduced in reality (e.g., post-explosion or flooding conditions), allowing the reinforcement learning model to learn effective strategies in complex and dynamic situations.

Architecture and Data Integration

The simulator operates in continuous interaction with two Extract components: the UDT, from which it retrieves user positions and characteristics, and the RL module, with which it exchanges data during training. Both modules share the same geographic data base derived from OpenStreetMap (OSM), processed to reflect the actual road network, including streets, bridges, and walkways. This ensures spatial consistency in movement simulations.

From the OSM topology, two primary elements are extracted:

- **Nodes**: intersections or key waypoints
- **Edges**: navigable connections between nodes

Edges are modeled to reflect real-world paths, including curves, detours, and physical constraints (e.g., canals or temporary obstacles).

User Modeling and Simulation

User positions during training are initialized within the simulator, enabling flexible and diverse testing configurations. Each virtual user is characterized by:

- **Geographical position** (latitude/longitude)
- **Heading** (direction of movement)

- **Speed** (average walking speed)

Geographical position is the essential parameter, while default values (e.g., 1.4 m/s for speed) can be assumed if other data are missing. All users are mapped to the corresponding OSM node or edge, ensuring alignment with RL's internal data structures.

Dynamic Risk Representation

The simulator also integrates dynamic information from the UDT regarding the safety status of different city areas. This includes:

- **Dangerous nodes**: areas to avoid due to hazards (e.g., fire, flood)
- **Safe nodes**: designated gathering points for evacuation

This risk-aware simulation framework enables robust and realistic RL training, supporting the development of adaptive evacuation strategies tailored to evolving emergency conditions.

Simulation flow

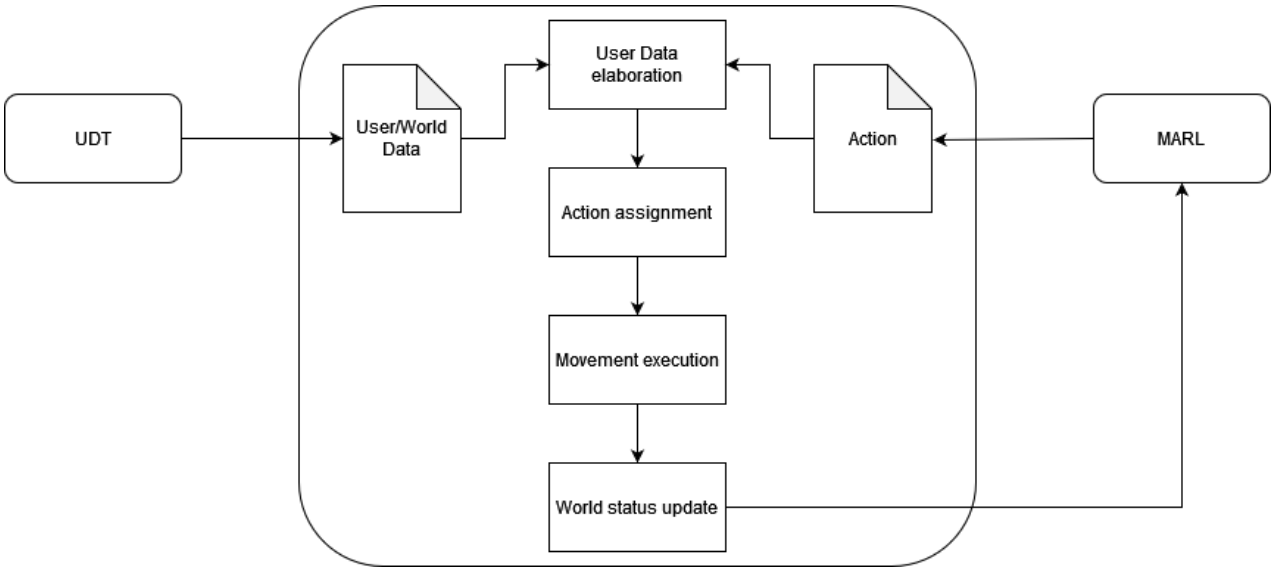

Figure 16 : *Simulation flow diagram of the PER use-case. It illustrates the interaction between the Urban Digital Twin (UDT), the Multi-Agent Reinforcement Learning (MARL) engine, and the simulation environment, encompassing user data elaboration, action assignment, movement execution, and world status updates.*

Once initialized, the simulator is ready to receive actions from the RL system, applying them to environmental and user data to initiate a trial-and-error loop for training the PER model. This process includes three main phases:

- Action Assignment: The simulator receives a set of desired user distributions across OSM nodes. It matches users' real-time positions—converted from GPS coordinates to OSM topology—with the actions, translating them into movement commands toward nearby safety nodes.

- Movement Execution: Each user moves toward their assigned destination, simulating real-world paths (e.g., roads and walkways in Venice) rather than straight lines. Each iteration represents one real-world second, ensuring realistic pacing and responsiveness to RLinstructions.
- World Status Update: After users move, the simulator updates the map with their new positions. It generates a 2D array indicating the number of people and available bandwidth per node and edge. This updated state is sent back to RL, which, if needed, produces new actions to start the next iteration.

Data exchange format

As input from the UDT the module needs a series of ingormation in order to start correctly the process: with LOGOS we settle a structured input data in JSON format, that contain the information organized in a JSON object. Each entry includes:

- timestamp: A time marker in ISO 8601 format (yyyy-mm-ddThh:mm:ss) indicating when the data snapshot was taken.
- static_data: A section reserved for configuration or context that does not change over time: the configuration of the environment (map) and the specific information over nodes and edges.
- dynamic_data: Contains the "live" information about the devices online and the actual status of the city:
  - device_count: Number of active devices detected and where those are dislocatede from a gegraphical point of view.
  - bandwidth: The real time status of network availability (bandwidth) expressed in terms of number of people
- dynamic_event_data: dedicate entry for the adverse event in progress. When the event is ongoing, the system is triggered to switch in emergency mode
- position_registry: A mapping of user or device positions, used for tracking movement and localization within the simulation or application environment.

This structured format ensures modular, real-time data handling for both monitoring and simulation purposes.

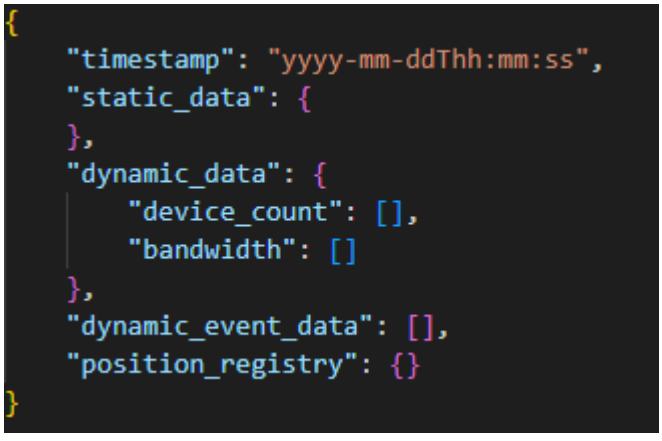

Figure 17 : *Structure of the JSON schema used in the PER use-case to collect and organize simulation data. The schema includes static and dynamic data fields, event information, and user position tracking, all timestamped to enable real-time analysis.*

For every instance of the simulation, the SIM provide as outcome a JSON object, formatted in a way so that the RL component can consume information in a streamlined and rapid manner.

The data are organized into several key components:

- positionRegistry: Stores the latest positions of users or devices, mapped by anonymous ID
- nodeInformation: Contains metadata about the nodes used in the map network for routing and localization
- edgeData: Describes the connections (edges) between nodes, used in the map network
- uniqueVector: internal decoding vector, which allows the identification between nodes/edges and the corresponding information about bandwidth and directions
- osmidCodeVector: A vector of OpenStreetMap (OSM) node IDs used to align simulation elements with real-world map data
- bandwidthVector: vector of information about the bandwidth available for the network used after the SIM execution

This structured output enables detailed post-processing, analysis, or visualization of both user movement and infrastructure behavior in complex urban environments.

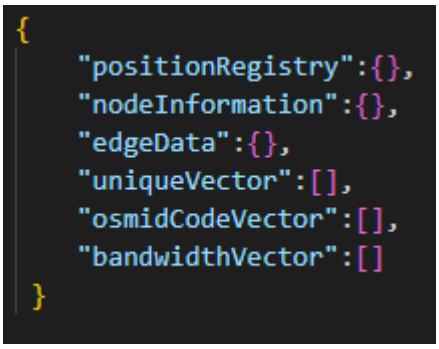

Figure 18 *: Data schema used to represent the simulation environment's structural components in the PER use-case. It includes information about node positions, edge connections, unique identifiers, OpenStreetMap codes, and bandwidth values, supporting graph-based modeling and analysis.*

4.2.7. Collaborative Framework: UDT, Simulation, and RL Agent Interaction

Upon activation of reinforcement learning for evacuation optimization, a collaborative framework comes into operation that brings together the key components of the PER use-case. This section outlines the mechanisms of interaction and communication protocols among the system's core elements, which collectively enable dynamic and adaptive response to urban emergencies in Venice.

4.2.7.1. Integration of Key Components

Seamless integration among the Urban Digital Twin (UDT), simulation environment, and reinforcement learning (RL) module is essential for real-time situational assessment and decision-making. The UDT functions as the central repository for both real-time and historical urban data, including the city's physical layout, environmental

conditions, and crowd dynamics. Drawing on this information, the simulator generates detailed models of individual and group behaviors under various emergency scenarios, creating a realistic testing ground. The RL component, in turn, leverages these simulations to evaluate and refine evacuation strategies in a controlled, risk-free environment.

4.2.7.2. Communication and Data Sharing Protocols

Communication across the system is managed via standardized data interchange protocols, primarily JSON, which ensure that all modules operate with consistent, up-to-date inputs. Updated city models from the UDT and behavioral data from the simulation feed into the learning engine, enabling continuous refinement of decision-making processes. Any changes in urban conditions or strategic objectives are promptly synchronized across the system, allowing for coordinated adaptation.

For inference, the trained RL model is deployed transparently via Sky Store Object Storage caching (see Deliverable D4.2), ensuring efficient propagation from training to deployment environments.

This collaborative framework ensures that the PER use-case remains responsive to the complexities of Venice's unique landscape. The tightly coupled interaction among components supports the development of effective, real-time evacuation strategies that reflect ongoing changes in both urban conditions and population movement patterns.

4.2.8. Operational Phases of the Reinforcement Learning Agent

Following the establishment of a collaborative framework among the UDT, simulation, and RL agent, focus shifts to the operational dynamics of the reinforcement learning agent itself. This section details the agent's two principal operational phases: inference and decision-making, and the training and learning process.

Inference and Decision-Making: During emergency scenarios, the RL agent engages in real-time inference and decision-making. Leveraging the latest urban data and simulation outcomes, the agent determines the optimal evacuation routes. This process involves analyzing current conditions, such as crowd densities and available pathways, to make split-second decisions that guide individuals to safety effectively. The goal is to maximize the efficiency of evacuation while minimizing risks to individuals affected by the emergency.

Training and Learning Process: The efficacy of the RL agent's decision-making is rooted in a comprehensive training and learning process. Utilizing historical data, simulated emergency scenarios, and feedback from previous evacuations, the agent undergoes extensive training phases. These phases, comprising between 500 to 2000 episodes with each episode involving 50 to 200 steps, allow the agent to explore various strategies and learn from the outcomes. This iterative process is essential for refining the agent's models, ensuring that the evacuation strategies evolve in line with changing urban dynamics and potential emergencies.

The subsequent section, will delve into the infrastructure deployment and latency considerations that underpin these operational phases, focusing on how cloud, HPC, and edge computing resources are orchestrated to support the RL agent's functions.

4.2.9. Infrastructure Deployment and Latency Considerations

The deployment of the PER Use-Case MVP infrastructure encompasses cloud and high-performance computing (HPC) resources, alongside edge computing capabilities, to support the comprehensive data processing and real-time decision-making demands of the system.

Cloud and HPC Infrastructure: The cloud infrastructure hosts the UDT and a segment of the simulation environment. This setup benefits from the cloud's scalable computing power to handle extensive urban data analysis and maintain the UDT's accuracy. Concurrently, HPC facilities, particularly at BSC, accommodate the reinforcement learning agent and part of the simulation that requires intensive computational resources. The HPC environment is pivotal for processing the complex algorithms and vast datasets integral to training the RL models, ensuring the system's predictive capabilities are both robust and efficient. UDT information is communicated transparently from cloud to HPC for RL training via object storage by leveraging Sky Store - see D4.2.

Edge Computing and Real-Time Data Processing: Edge computing infrastructure plays a critical role in minimizing latency for real-time data processing, especially during the operational deployment of RL models to guide evacuation efforts. Deploying trained models to edge devices enables the system to make swift decisions based on current urban conditions and individual locations. This approach ensures that evacuation instructions are timely and contextually relevant, addressing the immediacy of emergency scenarios.

In the edge, the system interacts with the phones of the people being evacuated in bi-directional manner to maintain up-to-date information. In one direction, phones continually send position information. This is used both for updating location for each evacuee, and for dynamically adjusting the edge service layout to minimize latency and/or power in response to incoming traffic indication, such as scaling the model serving in/out. In the opposite direction, model serving continually delivers up-to-date instructions to phones for further movement of evacuees until they reach safety.

The integration of cloud, HPC, and edge computing resources is carefully orchestrated to balance computational power with latency requirements. While the RL agent demands rapid updates for immediate decision-making, the UDT's updates, though less frequent, are vital for maintaining situational awareness. This layered infrastructure approach ensures that the system can respond to emergencies with precision, leveraging the strengths of each computing domain to support the evacuation process efficiently.

4.2.10. Evacuation Mobile App (EMA)

As part of the integrated emergency response system, a dedicated mobile application named **EMA (Evacuation Mobile App)** has been developed. EMA plays a central role in real-time user guidance and data collection during emergency scenarios, contributing to both situational awareness and the safety of individuals on the ground.

Technical Architecture

EMA has been implemented as a **hybrid mobile application**, combining **web-oriented technologies** with selected **native Android components**. This approach allows for rapid development and cross-platform compatibility while ensuring access to key native functionalities such as continuous geolocation tracking, background services, and network optimization (e.g., leveraging 5G connectivity when available).

The EMA mobile application has been developed using the **Apache Cordova** framework, a widely adopted solution for building cross-platform mobile apps using standard web technologies such as **HTML5, CSS3, and JavaScript**. Cordova enables the development team to write a single codebase while still accessing **native device functionalities** through a rich set of plugins. In the case of EMA, Cordova plugins are used to manage **geolocation**, **background execution**, **network state detection**, and **native notifications**.

This hybrid approach offers the advantage of **fast prototyping**, **easy updates**, and **broad compatibility**, while still supporting performance-critical features through native extensions when needed. Cordova's flexibility was essential in balancing the need for real-time responsiveness (e.g., during an emergency event) with maintainability and scalability across different Android device versions. Moreover, the framework allows seamless integration with backend APIs and data services, ensuring that the app can reliably exchange information with the UDT modules during both normal operation and emergency response phases.

The application is designed to operate seamlessly on Android smartphones and is optimized for both normal operating conditions and emergency states.

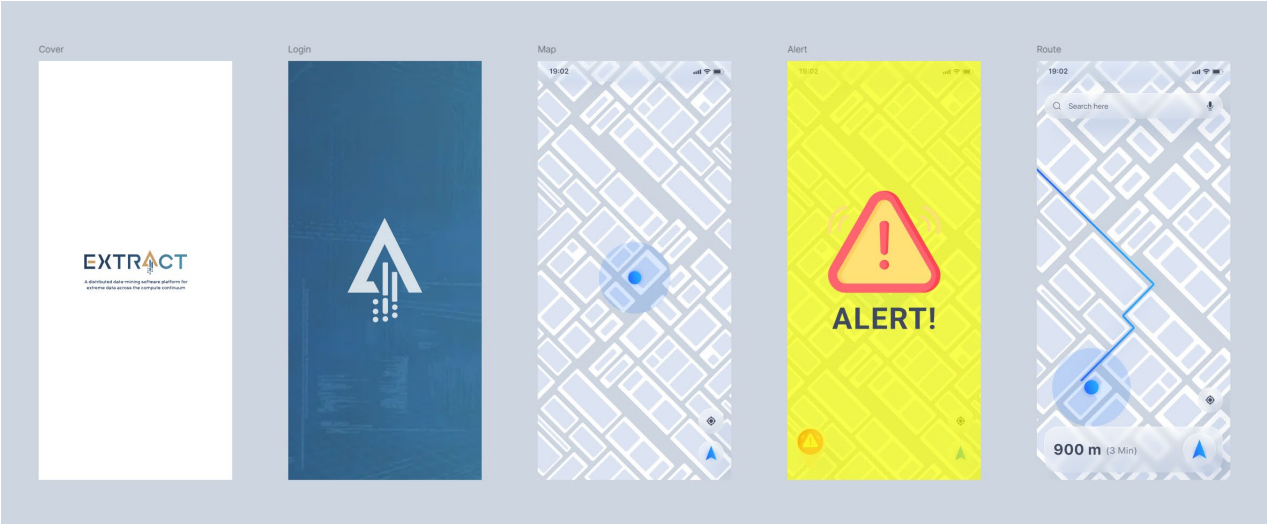

Figure 19 : *User interface mock-ups of the Evacuation Mobile App (EMA) developed for the PER use-case. The screens illustrate key functionalities including the splash screen, login, user location tracking, emergency alert display, and personalized evacuation route guidance.*

Core Functionalities

1. Continuous Geolocation Tracking

Under normal conditions, EMA continuously collects the **geographical position** of the user. The location data is gathered at regular intervals, with accuracy and frequency dynamically adjusted based on connectivity and battery optimization constraints. Whenever possible, **5G networks** are used to ensure low-latency, high-frequency updates, which are particularly valuable during fast-changing emergency conditions.

The high bandwidth, low latency, and network slicing capabilities of 5G make it ideally suited for mission-critical applications such as emergency management. In practice, the use of 5G significantly enhances the **responsiveness and reliability** of the app during evacuation scenarios, allowing for **high-frequency position updates**, near-instant delivery of evacuation instructions.

However, integrating 5G-based communication also presents **technical challenges**, especially given the heterogeneous availability of 5G infrastructure across urban areas. The app must be capable of **dynamically detecting and switching** between 5G, 4G, or Wi-Fi networks without disrupting the data flow or degrading the user experience. Moreover, handling **variable latency and handovers** between network types requires careful implementation of background services and connection management at the OS level.

While the **Apache Cordova** framework does not natively expose direct APIs for identifying the use of **5G (NR - New Radio)** networks, the EMA application overcomes this limitation by integrating **native Android plugins** to access advanced network information. Specifically, the app employs a **custom Cordova plugin** or extends existing ones (eg: cordova-plugin-network-information) to interface with the **Connectivity Manager** services. These native APIs allow the app to detect the current **radio access technology** in use and determine whether the device is connected via **5G**, **LTE**, **Wi-Fi**, or fallback technologies.

From a practical standpoint, this information is used to **optimize the frequency and granularity of geolocation data collection and transmission**. When a stable 5G connection is detected, the app can **increase the update rate** and support richer interactions with backend services due to the reduced latency and higher bandwidth. Conversely, when only 3G or unstable 4G connections are available, the app dynamically adjusts its behavior to preserve reliability and battery life.

Implementing this capability requires **bridging native code (Java/Kotlin)** with the Cordova JavaScript layer, maintaining full cross-platform flexibility while still leveraging device-level intelligence. Though slightly more complex, this hybrid strategy enables the EMA app to be **5G-aware**, a crucial feature for ensuring performance in real-time emergency scenarios.

Despite these challenges, the integration of 5G brings **tangible advantages**, including improved precision in user tracking, reduced communication delays, and enhanced scalability during mass emergency events where many users may be active simultaneously.

2. Privacy and data security

Throughout the development of the EMA application, the principles of **privacy by design and by default** have been strictly applied, in full compliance with the **General Data Protection Regulation (GDPR)**. From the earliest design phases, special

attention was given to minimizing the collection and processing of personal data. The app only acquires **geolocation data strictly necessary** for evacuation guidance and system coordination, and this data is collected **anonymously** without storing identifying user information.

All data transmissions between the app and the backend systems are encrypted using **industry-standard protocols (e.g., HTTPS/TLS)**. Location data is retained only for the duration necessary to ensure user safety during emergencies and is deleted or anonymized immediately afterward. Users are informed transparently about data processing practices through clear and accessible privacy notices, and explicit **user consent** is requested where required. Access to sensitive functionalities is limited via secure authentication mechanisms, and privacy impact assessments have guided the implementation of each component to ensure the **highest level of data protection by default**.

In line with the project's commitment to **data minimization** and **privacy compliance**, the EMA application implements a **zero-day data retention policy** for personal and location-related data. This means that **no user data is stored beyond the duration strictly necessary** to support real-time emergency response and evacuation guidance. As soon as the emergency event concludes or the session ends, all collected data — including geolocation coordinates and associated pseudonymous identifiers — is **automatically deleted** from both the device and backend systems. No historical user trajectories, identifiers, or personal data are retained for future use, analysis, or profiling. This policy ensures full alignment with the **"storage limitation" principle** under Article 5(1)(e) of the **GDPR**, reinforcing user trust and minimizing the risk of data breaches or misuse.

3. Emergency Trigger and User Guidance

When an emergency is detected—either through a broadcast alert or local event—the application transitions into **emergency mode**. In this state, EMA performs the following actions:

Receives evacuation instructions from the central system, including the best path to follow.

Displays a real-time evacuation route on the user's screen, guiding them step-by-step toward the nearest **designated safe zone**.

Updates the path dynamically if the situation evolves (e.g., a previously safe route becomes blocked).

The guidance interface is designed to be minimal, intuitive, and accessible, ensuring clarity even under stress.

5. Next Steps

TASKA use-case Final Release: Progress and Prospects

use-case A

Achievements:

- ***Successfully adapted a spikes detection neural network*** to detect and label specific spikes radio emission from the Sun in dynamic spectra.
- ***Real-time detection of Solar spikes***: the previous NN was successfully adapted in a real-time data flow in the NenuFAR receiver.
- ***Receiver Decision-making based on content***: detecting spikes trigger high resolution recording and data compression when the spikes are absent, leading to a 10x compression factor on the recorded data.

Ongoing Efforts:

- ***Validation of the code of use-case A2***: to develop a dedicated NN for detecting Jupiter burst emission.
- ***Integration in real-time data flow***: once the code is validated, A2 code can also be hosted inside the LaNewBa receiver and perform inference on real-time data.

Future Directions:

- ***Standardize the detection outputs***: find a Virtual Observatory compatible file (TFCat)
- ***Anomaly detection neural network***: to generalize the detection based on known morphology of emission but also capture the "unknown" emission.
- ***Detection error estimation***: to optimize and to improve the detection coverage and avoid Type-II error (false non-detection) on test data from simulation or archival data.
- ***Real-time telescope health monitoring***: assess the specification for anomaly detection applied to hardware in order to monitor the health of the antennas and possibly discarding bad antennas, in real-time, from the data combination.
- ***Distribution of detection alerts:*** to enable automatic triggering of piggy-back observations from decision-making (e.g. directed toward SETI receiver).

use-case C

Achievements:

- ***Data Processing Framework Refined***: Achieved workflow generality which enable the seamless composition of complex workflows for real-life applications.
- ***Looping workflow capability*:** to perform repetitive tasks (e.g. 100x time a single step with progressively variable parameters) to avoid manual insertion
- ***Data Management Improvements*:** testing the capability to access multiple S3 storage.
- ***Workflow state control*:** giving a status to each step and agency to user to resume/stop

Ongoing Efforts:

- ***Workflow logic insertion***: logic and decision can be implemented for branching the workflow to different directions. User is required to interact with

the user to resume or stop the workflow run. The current solution for parameterizing human interaction during operations is under development (through ssh, MQTT, etc.)

- ***Validation of heterogeneous computing resources***: to switch between Cloud computing and HPC computing in a seamless run for classical computation (Flagging, calibrating and imaging) to advanced methods (DDFacet/killMS) to lower the difficulty bar of mastering all the tools as well as showcasing new methods.
- **Decision-making in the workflow strategy**: depending on the size of the input dataset, specific reduction strategies must be implemented in the form of workflow templates (focused on I/O, computing, RAM) and parallelism/partitioning balance. The choice amongst several template must be done either by the end-user or based from previous extensive analysis of the parameter space. The partitionning feature of Lithops has to be flexible and smart enough to recommend a workflow path that minimizer the execution time.

Future Directions:

- ***Full Deployment on heterogeneous facilities***: various storage and computing capabilities and capability to control seamless data reduction on real-life observation (Solar data but not only). Contribution to the french SKA Regional Center.
- ***Implement priority modes*****:** to insert priorities between concurrent workflow runs between low-priority data reduction workflow and high-priority programs for which quick and reliable response time is required (e.g. transient detection and alert distribution).
- ***Low-energy impact optimization*****:** leveraging how CO2/WHr consumption can be efficiently monitor or capped depending on the extensive parameters space of small/large dataset and low/high computationnal needs.
- ***Dissemination effort of the workflow orchestration platform***: to demonstrate and implement similar solutions in other fields of astronomy and science.

use-case D

Achievements:

- ***Instrumental and observation simulator***: to generate accurate and realistic dataset for the training of the network.
- ***Adapting a neural network in the workflow***: to use the new imager instead of the classical one.
- ***General training testbed***: from the data simulation to the analysis of the metrics, Cassey provide a platform for benchmarking differents neural network, by taking away the burden of the radio interferometry data format or complex analysis tools.

Ongoing Efforts:

- ***Multiple training set production***: in order to improve the coverage of possible transient being present in the data, multiple ground truth variable sources are being simulated (unresolved point source, moving point source, varying extended emission and moving extended emission (just like Solar CMEs).

- ***Implementing transformers*:** as a new approach that better fits the "incomplete" and hollow nature of the visibilities measurements.
- ***Modular refactoring of the code***: preparing the code to future extensions and easy maintenance (other telescopes, "Clean code" principles)

Future Directions:

- ***Robust lightcurve error estimation**:* reconstructing the dynamic of variable radio source is the goal, but along with the uncertainty map associated with the uncertainties, which is rare to obtain in DL applications.
- ***Targeting non-imaging variable source detector:*** if the training rely solely on the data, in the fourier space, there is no longer need to go through imaging any longer. One could directly infer the presence, location and power of transient source directly in the visibilities. Such mode unlock the capability to analyze the real-time data flow in the correlator.
- ***Preparing for the A/E hybrid mode:*** which mixes use-case A detection (at high rate) and use-case D reconstruction (at low rate) using the complementarity of studying transient in the image plane and in dynamic spectra.

Final thought: The aim of TASKA is also to propose off-the-shelf solution for the upcoming needs and design of the SKA Regional Center, namely as a contribution to the French SRCnet. EXTRACT is a perfectly aligned demonstrator of this application and needs to be tested for scalability to SKA.

PER use-case Final Release: Progress and Prospects

Building on the MVP presented in D1.2, the PER use-case has evolved into a fully integrated and operational system. The focus of the final release lies in consolidating the platform's architecture, optimizing key components, and preparing for real-world validation within the urban environment of Venice.

Achievements

- ***Semantic UDT Integration*:** The UDT now ingests and harmonizes heterogeneous static and dynamic data sources in real time. Based on a semantic model built on SOSA/SSN and related ontologies, it supports accurate state representation and downstream decision-making.
- ***RL Advancements:*** The RL system has demonstrated the ability to outperform the baseline across different map sizes by evacuating people to safe points in the shortest possible time.
- ***Loop Closure Architecture*:** All core components—UDT, Simulator, RL, and EMA—are now functionally connected through a real-time, closed-loop architecture. Semantic alignment and robust data flows enable adaptive emergency response strategies.

Ongoing Efforts

- ***UDT Enrichment:*** Refinements are underway to enhance the UDT's urban knowledge base, improve dynamic state modeling, and extend real-world data integration. This includes additional microservices, expanded SPARQL queries, and updated dashboard visualizations.

- ***RL Optimization:*** Further refinement of reinforcement learning algorithms is necessary to improve accuracy, adaptability to complex evacuation scenarios, and performance on larger maps.
- ***EMA Testing and Volunteer Engagement:*** Testing activities involving trained volunteers (including local residents and retired emergency personnel) are ongoing. These trials support system robustness checks, UX evaluation, and continuous improvement of the Evacuation Mobile App (EMA).

Future Directions

- ***Real-World Pilot Execution:*** A live pilot in Venice is scheduled for October 2025. The test will evaluate system performance under realistic emergency conditions, focusing on responsiveness, route accuracy, and user experience.
- ***Community-Centric Preparedness*:** Through real-time personalized guidance, the system empowers individuals during emergencies, promoting trust, clarity, and community resilience.
- ***Sustainability and Transferability:*** The PER architecture—built on modular microservices and semantic web technologies—is designed for scalability and adaptability. Future deployments may extend to other cities facing similar evacuation challenges, including flood risk, mass tourism, or constrained infrastructure.
- ***Stakeholder and Policy Integration*:** Ongoing collaboration with civil protection authorities, local administrations, and infrastructure managers aims to align the system with urban resilience strategies and smart city policies. Outcomes may contribute to the definition of operational and policy frameworks for personalized evacuation solutions in Europe.

Final thought: The PER use-case aims to serve as a blueprint for next-generation, citizen-centric emergency management systems in smart cities. EXTRACT provides a demonstrator that is fully aligned with urban safety and digital twin strategies under development at municipal and European levels. The next steps will include testing the solution for scalability and operational integration in real-world deployments beyond Venice.

6. Conclusion

This deliverable marks the final release of the EXTRACT use-cases, consolidating the implementation, integration, and validation work carried out since Deliverables D1.1 and D1.2. It demonstrates the EXTRACT platform's capacity to support advanced data mining under extreme data conditions, leveraging a continuum of computing infrastructures from edge to HPC.

The PER use-case has progressed into a fully integrated system combining a semantic Urban Digital Twin, a multi-agent reinforcement learning engine, and a people movement simulation environment. The architecture has matured into a microservices-based deployment with real-time communication capabilities enabled by 5G MEC, supporting personalized evacuation strategies tailored to dynamic urban scenarios. This final configuration has undergone technical validation and is now prepared for operational testing in the city of Venice.

TASKA has evolved into a robust suite of sub-use-cases for real-time radio astronomy data processing. These include validated pipelines for solar event detection, modular workflow orchestration, and a novel deep learning-based imaging framework. All components adhere to FAIR and clean code principles, ensuring reproducibility, sustainability, and readiness for integration with broader open science infrastructures.

Throughout the development of both use-cases, several technical and integration challenges were encountered—ranging from coordinating heterogeneous components to ensuring real-time performance under constrained conditions. These challenges prompted valuable design iterations and contributed to a deeper understanding of workflow orchestration, semantic modeling, and cross-domain collaboration.

Looking ahead, areas for future improvement include performance optimization, scalability testing, and expanded validation in real-world scenarios. Additional work on usability, interoperability, and dynamic data integration will further enhance the platform's applicability in operational contexts.

Collectively, the results confirm the technical and scientific viability of the EXTRACT platform. The use-cases validate its core functionalities—scalable orchestration, semantic integration, low-latency inference, and trustworthy analytics across heterogeneous infrastructures. In doing so, they provide not only standalone demonstrators but also critical insights for final piloting activities and post-project exploitation strategies.

The EXTRACT use-cases—PER and TASKA—thus represent a milestone in the project's trajectory, offering mature, field-ready solutions that respond to urgent operational needs in emergency management and scientific research. The outcomes form a solid foundation for final-phase validation, stakeholder engagement, and long-term impact.

7. Acronyms and Abbreviations

- CNN – Convolutioonal Neural Network
- D – deliverable
- DDPG - Deep Deterministic Policy Gradient
- DoA – Description of Action (Annex 1 of the Grant Agreement)
- EC – European Commission
- EOSC - European Open Science Cloud
- EMA - Evacuation Mobile App
- HAAP - High Accuracy Positioning system
- HPC – High Performance Computing
- RLM - Reinforcement Learning Model
- MQTT - Message Queuing Telemetry Transport
- MS - Measurement Sets
- MVP - Minimum Viable Product
- NRAO - National Radio Astronomy Observatory
- ORN – Observatoire de Radioastronomy de Nançay
- OSM - Open Street Map

- OTN - Open Transport Network
- PER – Personal Evacuation Route
- PSF – Point Spread Function
- RDF - Resource Description Framework
- RFI - Radio Frequency interference
- RL - Reinforcement Learning
- SCR - Smart Control Room
- TASKA - Transient Astrophysics using the SKA pathfinder
- UDT – Urban Digital Twin

8. References

Akhaury U., Jablonka P., Starck J.-L., Courbin F., 2024, A&A, 688, A6. doi:10.1051/0004-6361/202449495

Chiche B. N., Girard J. N., Frontera-Pons J., Woiselle A., Starck J.-L., 2023, A&A, 675, A116. doi:10.1051/0004-6361/202245013