EXTRACT

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

# D1.2 First release of the EXTRACT use-cases

Version 1.0

Documentation Information

| | |
|---|---|
| **Contract Number** | **101093110** |
| **Project Website** | www.extract-project.eu |
| **Contractual Deadline** | 29.02.2024 |
| **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 |
| V0.3 | First Draft Chapter 4.2 PER Use-Case MVP |
| V0.4 | MVP Field Testing with Volunteer Engagement + Pictures/ Draft |
| V0.5 | PER Use Case MARL and TASKA Use-Case Breakdown |
| V0.6 | Minor additions for data propagation and model serving in the edge |
| V0.7 | TASKA MVP ObsParis inputs |
| V0.8 | TASKA update for use-cases A, D and E<br>Update PER MVP description and next steps<br>Conclusions |
| V0.9 | BSC Review |
| V1.0 | Submitted Version |

# Table of Contents

# 1. Executive Summary

This deliverable presents the first release of the PER (Personal Evacuation Route) and TASKA (Transient Astrophysics using the SKA pathfinder) use-cases.

The primary objective of this deliverable is to detail the progress and initial outcomes in moving from the definition of use cases to the actual deployment of Minimum Viable Products (MVPs).

The development of the PER and TASKA use-cases is grounded in the comprehensive requirements defined in the earlier phase of the project. These requirements were established through an in-depth analysis involving stakeholders' inputs, system actors' roles, and the anticipated application environments for the EXTRACT solution (see D1.1).

This process ensured that the foundation for the development of the EXTRACT platform/system was robust and tailored to real-world needs and challenges.

This deliverable outlines the evolution of the PER and TASKA use-cases from their conceptual stages to their current state.

It includes:

1. Detailed scenarios for each use-case,

2. an overview of the functionalities implemented,

3. the technical strategies employed

These aspects highlight the tangible progress made in translating the defined requirements into operational components of the EXTRACT project.

It's important to acknowledge that this first release of the MVPs is an initial step in an ongoing development process. The insights gained from this phase will fuel further refinements and enhancements in subsequent releases. The feedback from early testing and stakeholder engagement will be instrumental in fine-tuning the use-cases to better align with the evolving needs of the project.

## 1.1. Relation with other Deliverables and Tasks

This deliverable builds upon and extends the findings and developments reported in "From Lab to Market: The EXTRACT Use-cases" as delineated in Section 1.2.3 of the EXTRACT Description of Action (DoA). It specifically influences and is interlinked with key project components, driving advancements in subsequent tasks and deliverables.

Notably, this document has direct correlations with:

D2.2: First release of the EXTRACT data infrastructure and data mining framework: Enhancing data management capabilities and analytical processes essential for use-case optimization.

D3.2: First release of data-driven orchestration and monitoring: Implementing mechanisms for more effective data utilization and system oversight, integral to the iterative refinement of use-case functionalities.

D4.2: First release of the Compute Continuum and final integration plan: Contributing to the comprehensive integration strategy, ensuring cohesive system functionality across the computing spectrum.

This deliverable's findings and updates contribute significantly to the progression and alignment of these related tasks and deliverables, ensuring that the EXTRACT project maintains momentum towards its overarching objectives and key performance indicators.

## 1.2. Structure of the Document

This document is structured to provide a comprehensive understanding of the use-cases MVPs and their various components.

The Executive Summary offers a high-level overview and its relation to other deliverables and tasks.

Following this, Section 2 delves into detailing its application in extreme data scenarios within the PER and TASKA Use-Cases.

Section 3, Background and Motivation, includes detailed scenarios for the TASKA and PER Use-Cases.

Section 4 explores the technological backbone of the project. This part elucidates the technological framework, data management strategies, and workflow processes implemented in the TASKA and PER Use-Cases, highlighting the innovation and complexity of the use cases technical environment.

Section 5 outlines the Next Steps and Future Enhancements, charting the path forward for the project and envisioning its evolution.

The document concludes with Section 6, summarizing the project's outcomes and reinforcing its contributions to the field.

Finally, Sections 7 and 8 provide useful Acronyms and Abbreviations and References, respectively, aiding in the document's navigation and understanding.

# 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 that generate superior, actionable insights. Targeting the entire lifecycle of extreme data mining workflows, the EXTRACT platform prioritizes enhancements in performance, energy efficiency, scalability, and security while holistically addressing the aspects 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. The EXTRACT prototype is designed for real-time solar activity assessment, enabling further scientific exploration.

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.

# 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 SKA) working in the radio band, are producing a tremendous deluge of raw data collected from observation of the Universe. To achieve the objectives of the scientific program, data processing and product quality assessment are necessary before delivering the scientific products to the researchers. To reduce the burden of reducing the data "by hand", the scientific community requires the development of advanced tools 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, 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.

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, ii) implement new features in instrumental configuration automation, iii) enable manipulating scientific data in higher level of abstraction than regular scientific data management.

- **Data abstraction & user interaction**

The multiplicity of data files as well as their volume, slow down the efficiency to produce scientific results. The objective is to take away the burden of the data by considering them 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 interface is also key on top of the underlying machinery that handle the data.

- **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. Deliberate alteration or distortion of the data should be done with specific knowledge as part of data processing steps (e.g., data averaging, cleaning, calibration or transformation). No side process should impact artificially the quality of the measurement nor the reproducibility of a scientific product. 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: e.g., scientists would rather have a correct (but degraded) image rather than a sharper (but biased) image.

This principle should be maintained during the use of the technologies that are involved in TASKA.

- **Data Security**

The tools that will 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 teams. Compared to PER use case, no personal data is involved but sensitive information about the targets or the data processing recipes should stay within the perimeter of a given team until publicly released eventually.

- **Science reproducibility**

The dataset that undergoes data reduction will produce scientific products that can last for years as reference. They can also be compared to similar products created by 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.

### 3.1.2. Use Case A

The goal is 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.

A real-time pipeline close to the receiver (UnDysPuted) needs to be added to detect and classify events on the full-resolution data steam coming from the telescope (LaNewBa, left on Fig. 1), so as to enable the trigger of various data record resolutions (see Fig. 1) set upon science content. 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 is 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 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.

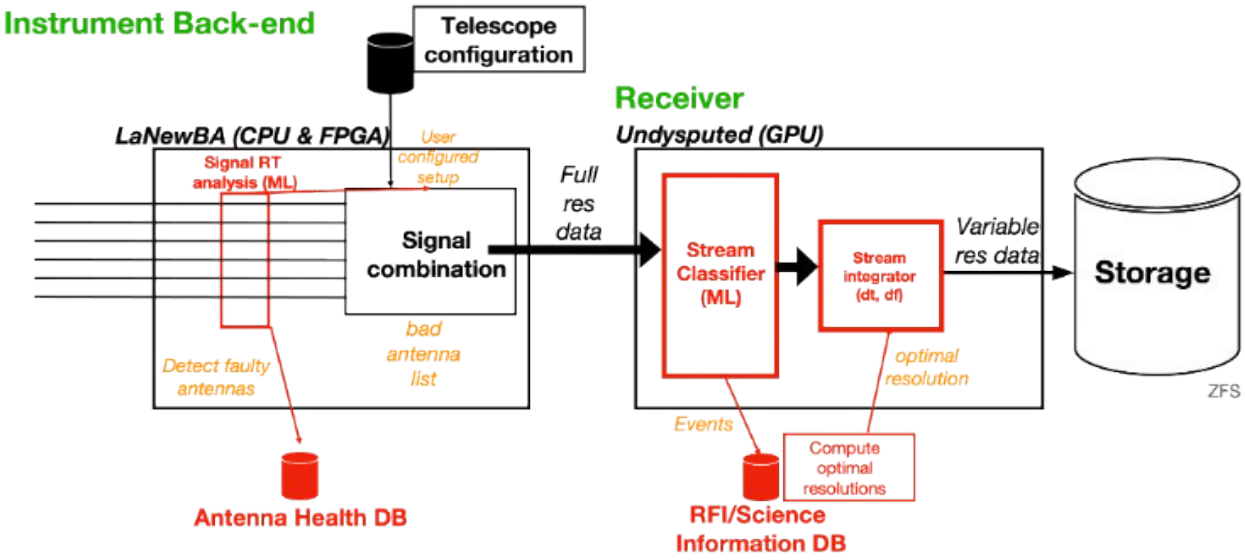



***Figure 1:*** *Use-case A: (Right) Automatic event detector and classifier. The output of the classifier will be logged and also serve for decision-making in the stream integrator down-stream. (Left) A similar system can detect faulty antennas and decide to weight them down before signal combination.*

This use case will not be covered by the current MVP since the precise definition of the required technologies is on-going.

### 3.1.3. Use Case C

The description covered in D1.1 has been studied and some tests have been conducted to build the architecture of the edge/cloud system which covers use case C. The main motivation for use case C is to provide to end-user 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, simplicity of implementation, processing time optimization. The performance of the platform (particularly for 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 use case will be central to the current MVP.

### 3.1.4. Use Case D

While TASKA use-case C focuses 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 dynamical radio sources, such as the Sun, such algorithms proved to be rather inefficient, mostly because the dynamical radio source overcomes the calibrators. We propose to develop a new imager, 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. Ideally, this new algorithm should operate directly on interferometric

visibilities (i.e., on the spatial Fourier domain), in order to reduce artifacts introduced by the regridding step of classical imaging pipelines.

This use-case will be 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.

This use case will not be covered by the current MVP since the precise definition of the required technologies is on-going.

### 3.1.5. Use Case E

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.

This use case will not be covered by the current MVP since the precise definition of the required technologies is on-going.

### 3.1.6. TASKA MVP

The TASKA MVP is designed out of TASKA Use-case C, proposing a workflow architecture for radio astronomy interferometric data imaging. A subset of three main tasks have been identified (flagging/rebinning, calibration and imaging). The MVP shall enable a dynamical composition of the three tasks, abstracting the data as objects.

## 3.2. PER Use-Case Scenario

The EXTRACT project targets critical challenges in emergency management within urban contexts, particularly focusing on the development and application of advanced data analytics and artificial intelligence technologies.

The selection of the PER use case is motivated by the pressing need to enhance urban resilience against disasters through innovative evacuation strategies.

This section elaborates on the rationale for choosing the PER scenario, underpinned by Venice's unique urban configuration and historical vulnerability to emergencies.

This discussion provides a further elaboration on themes introduced in D1.1, deepening the exploration of the use case's foundational elements.

### 3.2.1. Urban Challenges and the Need for Personalized Evacuation

Urban regions today face a spectrum of potential disasters, both natural and human-made. The complexity of these emergencies presents profound challenges for urban planning, engineering, and emergency services, highlighting the inadequacy of traditional, one-size-fits-all evacuation plans. These generic strategies often fail to consider the dynamic nature of urban environments and the diverse needs of their populations.

The first phase of the EXTRACT project underscored a critical demand for personalized evacuation strategies through an extensive stakeholder engagement process (see D1.1). This modern approach tailors evacuation instructions to individual circumstances, incorporating factors like personal mobility and environmental conditions.

Personalized plans signify a shift towards adaptable, efficient, and effective emergency management, facilitated by advancements in digital technologies such as Urban Digital Twins (UDTs).

These innovations offer detailed simulations of urban environments, enhancing the precision and relevance of evacuation planning.

### 3.2.2. The Concept of Personalization in Evacuation Planning

Evacuation planning's evolution towards personalization marks a pivotal shift in urban emergency management. This strategy offers evacuation instructions tailored to the unique circumstances of each individual, moving away from traditional, generic advisories.

Personalized evacuation planning is rooted in preparedness, flexibility, and adaptability—essential elements for community resilience. It enables strategic, real-time adjustments to evacuation plans, ensuring that responses evolve with the emergency situation and contribute to continuous improvement in disaster response mechanisms.

### 3.2.3. Factors Influencing Personalized Routes

The Personalized Evacuation Route (PER) use case adopts a comprehensive approach, considering various factors to optimize evacuation strategies:

**Individual Speed and Mobility:** Personal mobility characteristics are crucial in determining the most suitable evacuation route, ensuring plans are both realistic and actionable.

**Location and Proximity to Danger**: The individual's location relative to the emergency site dictates the urgency and direction of evacuation, aiming to quickly move people to safety.

**Crowd Density**: Addressing crowd density is essential to prevent bottlenecks and ensure smooth evacuation, with routes planned to reduce overcrowding.

The dynamics of crowd behaviour and movement depend on various logical crowd-related factors such as average speed, flow rate, volume and density of the crowd, as well as other psychological and social factors. Density is measured by the number of people per m2 and the flow rate is measured as meters per minute.
Static crowd density and moving/dynamic crowd density have risks and different limits. Crowd risk level can be determined by integrating crowd density and flow rate.
Crowd risk increases with density and flow rate, and moves into high risk when the density exceeds a certain threshold, e.g. five-person/m2 (suggested by Still, 2011).
"Overcrowding" is the frequent cause of crowd disasters. Thus, "high crowd density" can lead to serious safety issues and crowd disasters. When the flow reaches critical levels of congestion, crowd motion transitions into a "stop-and-go" pattern and causes a phenomenon called "crowd turbulence". Turbulent crowd motion is characterised by random, unintended displacements of groups in all possible directions (mass motion), which trigger disasters. Large gatherings of pedestrians are found in closed facilities such as shopping malls, stadiums, train stations and in open facilities such as walkways and parks. The number and severity of tragedy crowd disasters in high-density public events have risen significantly in the past decade.

**Expertise of the people:** The overall number of inhabitants in the city of Venice has been decreasing since many years and it has recently fallen below the threshold of 50.000 dwellers. On the other hand, tourists are a constant phenomenon in the city and the number of presences (i.e. the number of nights spent in Venice by visitors) are almost 11 millions[i]. The trend has been positive since 2013 with the only exception of 2020 due to the COVID-19 pandemic. In the context of evacuation planning this has a precise meaning that cannot be ignored: in case of an emergency it is more likely to be dealing with someone who doesn't know the city and its peculiarities, than evacuating people who know perfectly how to move inside the city. This translates into a higher probability of congestion, especially where the narrow pathways and bridges create dangerous bottlenecks.

**Environmental and Infrastructural Conditions**: Real-time adaptation of evacuation routes considers current environmental and infrastructural challenges, ensuring navigational feasibility.

This approach demonstrates the practical application of personalized evacuation, highlighting the role of technology in enhancing emergency response effectiveness and community safety.

By focusing on the unique needs of individuals, the PER use case illustrates a significant advancement in disaster management, showcasing a commitment to leveraging technology for societal benefit.

### 3.2.4. Rationale for Choosing Venice and the Fire Scenario for PER Use case

Venice, with its unique urban characteristics, including aging infrastructure, narrow pathways, and reliance on waterways, epitomizes the challenges of emergency evacuation planning in dense urban environments. This complexity, underscored by the city's susceptibility to emergencies like fire, floods, and overcrowding, mandates a sophisticated approach to ensuring public safety.

Hence, Venice was selected as the testbed for the PER use case, aiming to harness these distinctive challenges to develop and refine advanced evacuation strategies.

The choice of a fire scenario for testing within the PER use case emphasizes the critical need to address the immediate and tangible threats posed by such disasters, drawing lessons from historical incidents like the Fenice opera house fire (see also D1.1).

Incorporating crowd dynamics, grounded in crowd science, is vital to customizing evacuation plans to Venice's unique urban conditions, particularly in managing crowd density and movement during emergencies.

### 3.2.5. Preparatory Field Testing with Volunteer Engagement

On the 2nd of February in 2024, the PER use case underwent field testing in Venice, on San Servolo island, focusing on the evaluation of its MVP against the backdrop of personalized evacuation strategies.

This exercise harnessed the expertise of retired police officers, selected for their comprehensive experience in operational scenarios, to create a testing environment reflective of real-world emergency evacuation conditions.

The selection of these volunteers, with their profound knowledge of crowd management, facilitated an in-depth examination of evacuation dynamics, particularly the "herding" effect and how individuals' movement is affected by obstructions.

The test was methodically conducted, with volunteers submitting detailed information about themselves to aid in customizing evacuation routes.

This process underscored the PER use case's emphasis on adapting evacuation strategies to individual speeds and crowd density considerations.

Enrico Lando, a director specializing in documentary filmmaking, alongside a retired officer with extensive background from the Venice Police Headquarters, oversaw the documentation of this exercise, ensuring a thorough and analytical record of the test's proceedings.

This field test showcased the significant impact of crowd density on evacuation efficacy and safety, reinforcing the importance of personalized evacuation planning in urban environments, particularly those frequented by tourists.

The exercise also noted the essential role of emergency services in guiding and controlling the evacuation process.

In synthesizing these observations, the field test validated the PER use case MVP's approach to improving emergency management within urban settings through personalized evacuation routes.

The involvement of individuals with specialized knowledge and backgrounds contributed to a nuanced understanding of the practical challenges and solutions in emergency evacuations, aligning with the project's objective to enhance urban safety and resilience through tailored and flexible response strategies.


EXTRACT
Finanziato dall'Unione europea
Città metropolitana di Venezia
MI 10T LITE | GIBO


*Volunteers Participating in the Preparatory Field Testing on San Servolo Island, Venice*

# 4. MVP Technologies and Data Infrastructure

This section describes the development, implementation, and preliminary testing of the MVP for the TASKA and P.E.R. use cases.

It provides details on the achieved objectives, the technologies used, and the initial results.

The primary objective of the MVPs is to exhibit selected functionalities within more straightforward scenarios or applications than those anticipated for the final use cases by the project's end. This approach allows for initial validation and refinement of the EXTRACT platform's capabilities in addressing extreme data challenges.

This approach aims to verify that the system's components interact effectively and that they are operational for the simpler scenarios presented by the MVP, rather than for the full-scale complexity of extreme data scenarios anticipated in the final use cases.

## 4.1. TASKA MVP

### 4.1.1. Overview of Requirements, Datasets and Technologies for the TASKA MVP

The requirements for use case C reside in creating a dynamic framework to enable a user (the 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 main contribution for TASKA-C is the capability to manipulate and process the data as “objects” and manipulate it as a single abstract entity independently of the size, number or nature of the data files. The user expects to have a high-level vision over the workflow by not being cluttered with manual data logistics.

**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, as well as stopping, resuming, restarting tasks with other set of parameters.

**Workflow data robustness and flexibility**

The data sets are presented in the Cloud part as a set of objects that are visible and can be processed composing the tasks available in the MVP. Datasets can vary in volume by several orders of magnitude (from 100s of MBs to several TBs) and the workflow should ensure a minimal reduction time. Reduction time will be optimized for data partitioning, parallel processing, decision making regarding the optimal distribution of computation between workers.

**Workflow user interaction**

User interaction includes defining the workflow by arranging the task sequence, the input/output, inspection of the data & results and the use of a user script. Users should be able to seamlessly access the various outputs. Users can interact with the workflow through a Python notebook and possibly a dedicated interface for the design or accessing the results.

The results should be attached to 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).

The figure below describes in more detail the structure of the use case C. On the edge part, the data are produced and are put in a dedicated format (Measurement Sets, "MS" format). 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 4.1.2). Data ingestion will organise and store the dataset and produce the relevant meta-data. The user should use a predefined data reduction recipe or design its own workflow in the limits defined by the current MVP (see. 4.1.3).

The list of available tasks can evolve and will be fit to the framework defined in the MVP.

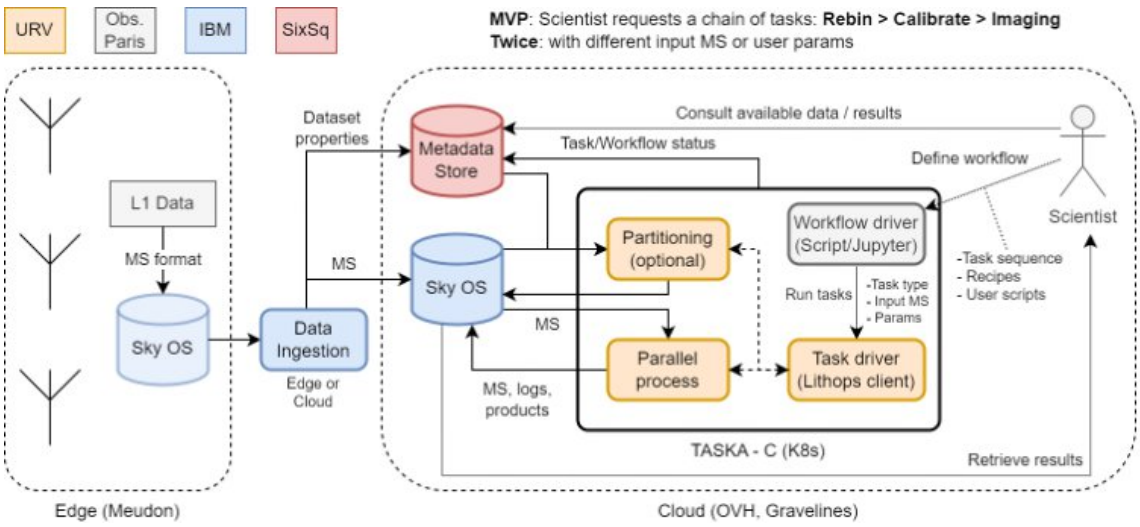



***Figure 2:*** *Main view of the edge/cloud interaction system for data reduction orchestration in use-case C.*

### 4.1.2: Data Management

**Description of use Case C data**

The specificities of Use Case C is to deal with Measurements Sets data produced by a radio interferometer. By nature, this data cannot be used directly and needs to undergo specific mandatory steps before being scientifically used. Each input and output are defined for each task for the MVP.

The input data follows the Measurement Sets (MS, NRAO [REF]) standard, a specific data format hosting interferometric data. This standard contains the measured data from a target, as well as ancillary data which give context to the observation. These data are manipulated through specific data management tools to open, write, split, merge the MS files while maintaining the data file integrity. A given MS file can be transformed and grow or shrink in size depending on the task applied to it. A given dataset is a series of joint MS files that build a coherent set of measurements. Each MS represents a slice of the observation in the frequency domain. It contains a time axis, a frequency axis, information about the measurement antennas, their status, and dedicated tables addressing the quality of the data.

Figure 2. describes in a generic way the chaining of data reduction steps. Each individual step represents an action taken on the data. Input data undergoes a transformation defined by the user parameters, and produces results as well as altering the data in preparation for the next step. Subsequent steps could require human intervention through a user-defined script (e.g. compute a physical number from products of step [n], mandatory to define the input user parameters of step [n+1]). It is possible that the intermediate steps can be automatically handled without human intervention (e.g. computing the mean value of a product, passing it as input parameter to the next step).

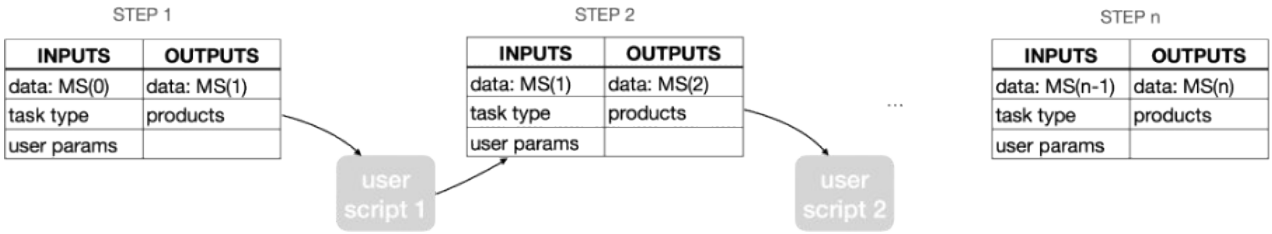



***Figure 3:*** *Chain of steps in the workflow. Each step is generically defined by a task type, input data, output data, input parameters and products. Each task can be interleaved with specific user script to prepare the next data reduction step.*

**The "Measurement Set" format**

The Measurement Set format has been defined by the National Radio Astronomy Observatory (NRAO) as a container that hosts binary tables and subdirectories (or subtables). The main binary table contains the scientific measurements and the subtables contain meta-data about the observation (e.g. Antenna list, pointing, history of the MS file).

One should ensure the data integrity all along the data processing. Binary tables are not directly accessible and dedicated libraries are required to open the MS.

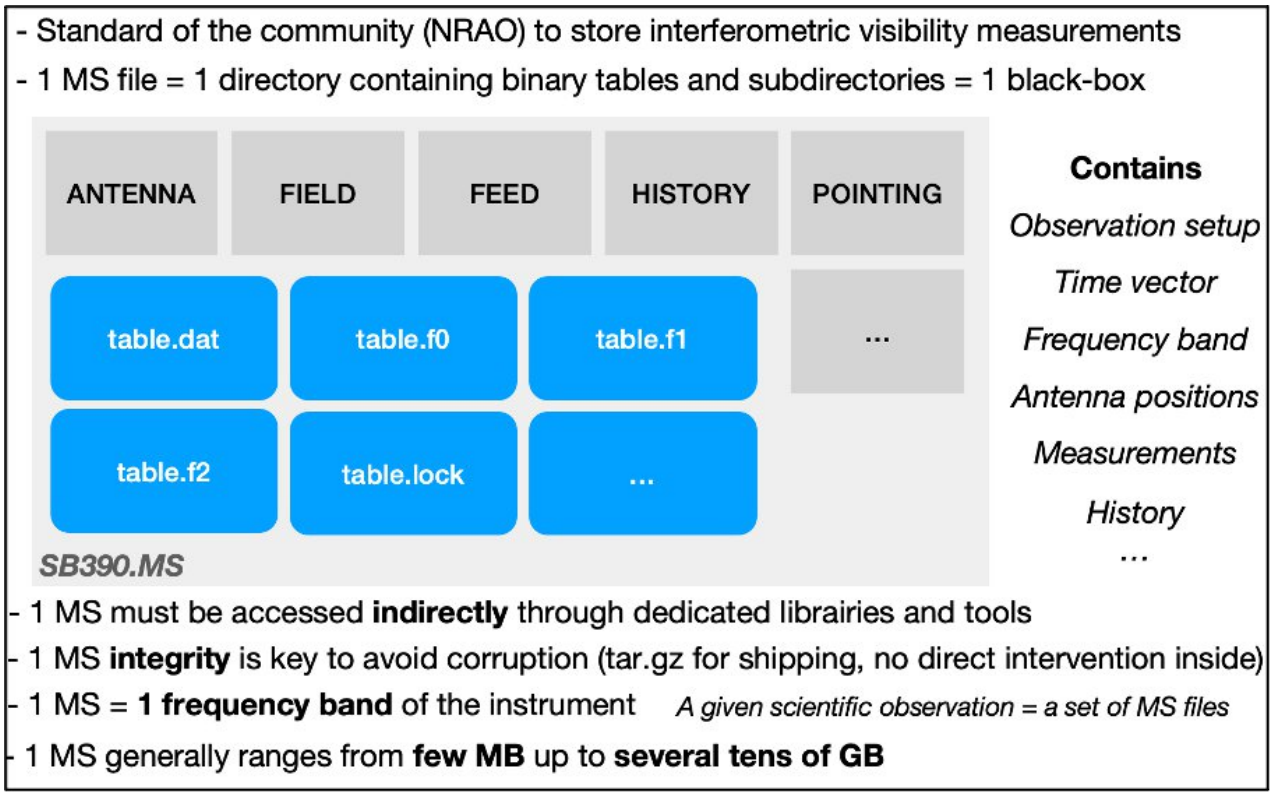



***Figure 4:*** *Representation of the structure of a Measurement Set*

**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 build complementary meta-data. These 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 the MVP**

The MVP for Use Case C includes small datasets and mid-size datasets that are representative of the radio astronomy applications covered by EXTRACT. It is composed of a set of data containing static continuum data ("Cygnus loop"), a dataset potentially containing variable radiosource in a field of constant sources, an observation of the Sun which dominates the field and the data. It is transient by nature. It comes with the associated calibration observations of Use Case A.

The main target for the MVP is the capability to reduce a whole slot of observation conducted for the NenuFAR Survey. The hundreds of pointings cannot be reduced by hand and should be reduced automatically.
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).
Future versions of the use case C will address bigger datasets (1.4 TB) spanning over longer observing times (630min). These datasets will be challenging to reduce and should take advantage of the flexibility and data fractioning provided by the infrastructure.

| EXTRACT? | Short ID<br>*string* | Short name<br>*string* | OBSID<br>*[YYYYMMDD_HHMMSS_YYYYMMDD_HHMMSS_STRING]* | end time<br>*UT* | Obs length<br>*min* | Total Volume<br>*GB* | # of data Unit<br>*integer* |
|---|---|---|---|---|---|---|---|
| | | IMAGING DATASETS | | | | | |
| YES | VIS-CYGLOOP | Cygnus Loop | 20220308_091800_20220308_111300_J2000_TRACKING | 11:13:00 | 115 | 740 | 156 |
| YES | | Cygnus X-3 (part 1) | 20220615_221500_20220616_020200_CYGNUS_X-3_IAN | 2:02:00 | 227 | 561 | 244 |
| YES | | Cygnus X-3 (part 2) | 20220616_020200_20220616_060000_CYGNUS_X-3_IAN | 6:00:00 | 238 | 589 | 244 |
| YES | | Sun | 20230411_101000_20230411_134000_SUN_TRACKING | 13:40:00 | 210 | 16 | 40 |
| YES | | Cas A calibration 1 | 20230411_100000_20230411_101000_CAS_A_TRACKING | 10:10:00 | 10 | 2.7 | 40 |
| YES | | Cas A calibration 2 | 20230411_134000_20230411_140000_CAS_A_TRACKING | 14:00:00 | 20 | 5.6 | 40 |
| | | | | | | | |
| YES | | Survey S1_SL37_CELL_4_RUN_1 | 20230328_230000_20230328_232800_S1_SL37_CELL_4_RUN_1 | 23:28:00 | 28 | 29 | 64 |
| YES | | Survey S1_SL37_CELL_128_RUN_2 | 20230328_232800_20230328_235600_S1_SL37_CELL_128_RUN_2 | 23:56:00 | 28 | 29 | 64 |
| YES | | Survey S1_SL37_CELL_94_RUN_6 | 20230328_235600_20230329_002400_S1_SL37_CELL_94_RUN_6 | 0:24:00 | 28 | 29 | 64 |
| YES | | Survey S1_SL37_CELL_95_RUN_5 | 20230329_002400_20230329_005200_S1_SL37_CELL_95_RUN_5 | 0:52:00 | 28 | 29 | 64 |
| YES | | Survey S1_SL37_CELL_128_RUN_5 | 20230329_005200_20230329_012000_S1_SL37_CELL_128_RUN_5 | 1:20:00 | 28 | 29 | 64 |
| YES | | Survey S1_SL37_CELL_79_RUN_6 | 20230329_012000_20230329_014800_S1_SL37_CELL_79_RUN_6 | 1:48:00 | 28 | 29 | 64 |
| YES | | Survey S1_SL37_CELL_82_RUN_3 | 20230329_014800_20230329_021600_S1_SL37_CELL_82_RUN_3 | 2:16:00 | 28 | 29 | 64 |
| YES | | Survey S1_SL37_CELL_7_RUN_2 | 20230329_021600_20230329_024400_S1_SL37_CELL_7_RUN_2 | 2:44:00 | 28 | 29 | 64 |
| YES | MVP | Survey S1_SL37_CYG_A | 20230329_024400_20230329_030000_S1_SL37_CYG_A | 3:00:00 | 16 | 16 | 64 |
| | | | | | | | |
| Optional* | | NCP Cosmic Dawn | 20230321_180000_20230322_043000_NCP_COSMIC_DAWN | 4:30:00 | 630 | 1393 | 244 |
| Optional* | | Cyg A calibration | 20230322_043000_20230322_050000_CYGA_COSMIC_DAWN | 5:00:00 | 30 | 62 | 244 |
| | | | | | | | |
| *non standard recipe | | | | | | | |

***Table 1:*** *Available data for the MVP development and validation are in the red box. Future evolutions from the MVP include larger and more complex datasets*

## 4.1.3: 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.

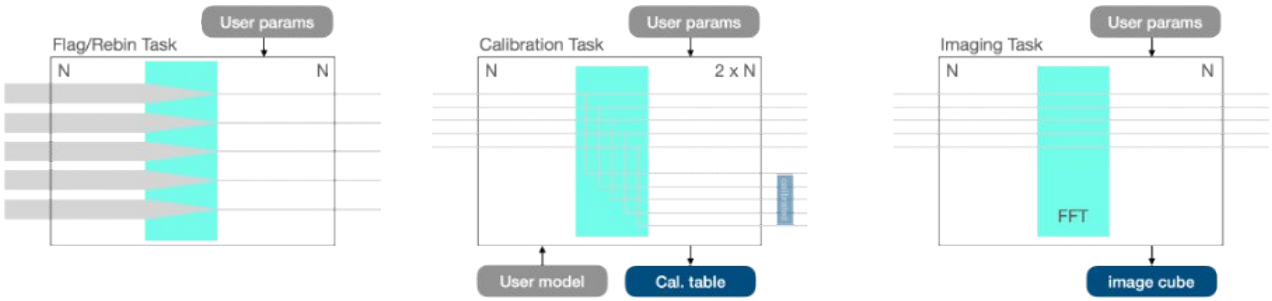



***Figure 5:*** *The three main tasks planned for the MVP: Flag/Rebin task, Calibration task and Imaging task.*

- **Flagging/Rebinning Task**

**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**

**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**

**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.

### 4.1.4: Workflow Example

The following information provides a sample data reduction from start to end with the series of commands and parameters.

**Step 1: Flagging and rebinning the data**
Command principle:
- For all MS in DATA, run sequentially the flagging and rebinning steps to create a lossy scientific compression of the data.
- Output data are MS files with a degraded resolution and stored in DATAREB.

Inputs/Outputs:
- Input Data: DATA/SB*.MS
- Input parameters:
  - STEP1-flagrebin.parset
  - STEP1-NenuFAR64CS1.lua
- Output Data: DATAREB/SB*.MS

Run bash command for a single MS:
DP3 PARAMETERS/STEP1-flagrebin.parset msin=SB205.MS
msout=DATAREB/SB205.MS

**Step 2a: Calibration solutions computation**
Command principle:
- For all MS in DATAREB, solve the calibration optimization to deduce instrumental solutions in an embarrassingly parallel way.
- Output data are HDF5 tables containing calibration solutions to apply to the data.

Inputs/Outputs:

- Input Data: DATAREB/SB*.MS
- Input parameters:
  - PARAMETERS/STEP2A-apparent.sourcedb
  - PARAMETERS/STEP2A-calibration.parset
- Output Data:
  - OUTPUT/SB*.h5 (Format HDF5)

Run bash command for a single MS:
DP3 PARAMETERS/STEP2A-calibration.parset msin=DATAREB/SB205.MS cal.h5parm=OUTPUT/SB205.h5 cal.sourcedb=PARAMETERS/STEP2A-apparent.sourcedb

**Step 2b: Subtracting strong sources**

Command principle:

- For all MS in DATAREB, subtract the strong sources given a description of the sky.
- Output is a new column SUBTRACTED_DATA in each MS

Inputs/Outputs:

- Input Data:
  - DATAREB/SB*.MS
  - OUTPUT/SB*.h5
- Input parameters:
  - PARAMETERS/STEP2A-apparent.sourcedb (introduced in STEP2A)
  - PARAMETERS/STEP2B-subtract.parset
- Output Data:
  - Adding a column inside SB*.MS

Run bash command for a single MS:
DP3 PARAMETERS/STEP2B-subtract.parset msin=DATAREB/SB205.MS sub.applycal.parmdb=OUTPUT/SB205.h5 sub.sourcedb=PARAMETERS/STEP2A-apparent.sourcedb

**Step 2c: Applying calibration solutions to data**

Command principle:

- For all MS in DATAREB, apply the calibration solution
- Output is a new column CORRECTED_DATA in each MS

Inputs/Outputs:

- Input Data:
  - DATAREB/SB*.MS
  - OUTPUT/SB*.h5
- Input parameters:
  - PARAMETERS/STEP2C-applycal.parset
- Output Data:
  - Adding column inside SB*.MS

Run bash command for a single MS:
DP3 PARAMETERS/STEP2C-applycal.parset msin=DATAREB/SB205.MS apply.parmdb=OUTPUT/SB205.h5

**Step 3: Imaging data**

Command principle:

- For all MS in DATAREB, read and use the CORRECTED_DATA columns from all MS.

- The output is a single image in FITS format containing spatial information in astronomical coordinates.

Inputs/Outputs:

- Input Data:
  - DATAREB/SB*.MS
- Output Data:
  - Single image (Format FITS) in OUTPUT directory

Run bash command:

wsclean -size 1024 1024 -pol I -scale 5arcmin -niter 100000 -gain 0.1 -mgain 0.6 -auto-mask 5 -local-rms -multiscale -no-update-model-required -make-psf -auto-threshold 3 -weight briggs 0 -data-column CORRECTED_DATA -nmiter 0 -name OUTPUT/Cygloop-205-210-b0-1024 DATAREB/SB{205..210}.MS

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

**Parameters list for Cygnus Loop Recipe**

| Step number | Step name | Parameter name | Parameter value | Comment |
|---|---|---|---|---|
| 1 | **RebinningStep** | | | |
| | *Example bash command* | *DP3 PARAMETERS/STEP1-flagrebin.parset msin=SB205.MS msout=DATAREB/SB205.MS* | | |
| | | steps | [aoflag,avg,count] | *list of steps in this rebinning step, name can be anything but some words are reserved (e.g. count). Step parameters are the stepname.param = valu* |
| | | aoflag.type | aoflagger | *name of algorithm used* |
| *(only relevant parameters)* | **STEP1-flagrebin.parset** | aoflag.strategy | /ayman-extract/parameters/rebinning/STEP1-NenuFAR64C1S.lua | *strategy file (~1 per entire radiotelescope)* |
| | | avg.type | averager | *averaging the data (lossy compression)* |
| | | avg.freqstep | 4 | *average stacks of 4 frequency steps 4 channels -> 1 channel* |
| | | avg.timestep | 8 | *average stacks of 8 time steps: 8 time steps --> 1 time step* |
| 2A | **CalibrationStep** | | | |
| | *Example bash command* | *DP3 PARAMETERS/STEP2A-calibration.parset msin=DATAREB/SB205.MS cal.h5parm=OUTPUT/SB205.h5 cal.sourcedb=PARAMETERS/STEP2A-apparent.sourcedb* | | |
| | | msin.datacolumn | DATA | *Input data containing visibilities* |
| | | steps | [cal] | *List of steps of the calibration step (only one here)* |
| | | cal.type | ddecal | *Type of calibration: "dde" = direction-dependent effect or "gaincal" = gain calibration* |
| | | cal.mode | diagonal | *Values can be: calar, scalarphase, scalaramplitude, diagonal, diagonalphase, diagonalamplitude, fulljones, 'tec' and 'tecandphase'* |
| | **STEP2A-calibration.parset** | cal.sourcedb | /ayman-extract/parameters/calibration/STEP2A-apparent.sourcedb | *reference to calibration model (PROVIDED)* |
| | | cal.h5parm | "" | *name of the output solution file after calibration (hdf5 format)* |
| | | cal.solint | 4 | *Number of time steps for computing 1 solution (here 4 time steps ==> 1 solution), 4000 time steps ==> 1000 solutions* |
| | | cal.nchan | 4 | *Number of frequency channels to compute 1 solution* |
| | | cal.maxiter | 50 | *Number of iteration for computing 1 solution (less iteration means less quality or less convergence)* |
| | | cal.uvlambdamin | 5 | *minimal UV range for computing the solution. A selection on UV values can sometimes helps the convergence* |
| | | cal.smoothnessconstraint | 2,00E+06 | *Smoothness factor imposed on the solutions for regularity* |
| 2B | **SubtractionStep** | | | |
| | *Example bash command* | *DP3 PARAMETERS/STEP2B-subtract.parset msin=DATAREB/SB205.MS sub.applycal.parmdb=OUTPUT/SB205.h5 sub.sourcedb=PARAMETERS/STEP2A-apparent.sourcedb* | | |
| | | msin.datacolumn | DATA | *Input data containing visibilities* |
| | | msout.datacolumn | SUBTRACTED_DATA | *Target column after subtraction part* |
| | | steps | [sub] | *List of steps for the subtraction step (only one here)* |
| | | sub.type | "h5parmpredict" | *name of the task (predict the data from some reference sources (sub.directions) and simulate data before subtraction)* |
| | **STEP2B-subtract.parset** | sub.sourcedb | /ayman-extract/parameters/calibration/STEP2A-apparent.sourcedb | *reference to calibration model (PROVIDED)* |
| | | sub.directions | [[CygA],[CasA]] | *Name of the sources to subtract (these are really powerful and pollute the data)* |
| | | sub.applycal.parmdb | "" | *name of the output solution file after calibration (hdf5 format)* |
| | | sub.applycal.steps | [sub_apply_amp,sub_apply_phase] | *name of substeps for "applycal" - apply solution in amplitude, then in phase* |
| | | sub.applycal.correction | fulljones | *Same as cal.mode* |
| | | sub.applycal.sub_apply_amp.correction | amplitude000 | *reference to the table header name in the hdf5 file* |
| | | sub.applycal.sub_apply_phase.correction | phase000 | *reference to the table header name in the hdf5 file* |
| 2C | **ApplyCalibrationStep** | | | |
| | *Example bash command* | *DP3 PARAMETERS/STEP2C-applycal.parset msin=DATAREB/SB205.MS apply.parmdb=OUTPUT/SB205.h5* | | |
| | | msin.datacolumn | SUBTRACTED_DATA | *Input data containing visibilities (subtracted)* |
| | | msout.datacolumn | CORRECTED_DATA | *Output column after calibration* |
| | | steps | [apply] | *List of steps for the apply step (only one here)* |
| | | apply.type | applycal | *Type of calibration: "applycal"* |
| | **STEP2C-applycal.parset** | apply.steps | [apply_amp,apply_phase] | *same as subtraction step* |
| | | apply.apply_amp.correction | amplitude000 | *same as subtraction step* |
| | | apply.apply_phase.correction | phase000 | *same as subtraction step* |
| | | apply.direction | [Main] | *Application to the main MS direction* |
| | | apply.parmdb | "" | *Reference to the output solution file after calibration (hdf5 format)* |
| 3 | **ImagingStep** | | | |
| | Example bash command | wsclean -size 1024 1024 -pol I -scale 5arcmin -niter 100000 -gain 0.1 -mgain 0.6 -auto-mask 5 -local-rms -multiscale -no-update-model-required -make-psf -auto-threshold 3 -weight briggs 0 -data-column CORRECTED_DATA -nmiter 0 -name OUTPUT/Cygloop-2( | | |
| | | data-column | CORRECTED_DATA | *Data colum to image* |
| | | size | "1024 1024" | *Image size in pixel* |
| | | pol | I | *Polarization to image* |
| | | scale | 5arcmin | *Size of a pixel on the sky* |
| | | niter | 100000 | *Number of iterations for the CLEAN deconvolution algorithm (Minor loops)* |
| | | nmiter | 0 | *Number of iterations for the CLEAN deconvolution algorithm (Major loops)* |
| | | gain | 0.8 | *CLEAN gain (minor loops)* |
| | | mgain | 0.6 | *CLEAN gain (major loops)* |
| | | auto-mask | 5 | *Imaging parameter for masking* |
| | | auto-threshold | 3 | *Self-determination of the threshold* |
| | | local-rms | 1 | *Option* |
| | | multiscale | 1 | *Activate multiscale* |
| | | no-update-model-required | 1 | *Option* |
| | | make-psf | 1 | *Ouput the instrument impulse response* |
| | | weight | "briggs 0" | *CLEAN weights* |

***Table 2:*** *Example parameters required for Cygnus Loop recipe chaining various flavors of the rebin, calibration and imaging tasks.*

### 4.1.5: 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 use 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.

**Workflow composition**

The available steps in the MVP are Rebinning, Calibration and Imaging, and they can be composed as needed. The MVP performs a sequence of the three steps (see Figure 6, with outputs of a step directly feeding the next one to finally produce scientifically valuable products (as further detailed in Section 4.1.1). Each step is configured with specific parameters, which include the input and output paths to data as well as the desired operation parameters. The desired operation parameters can be modified from the workflow definition, dynamically adjusting them based on the outcomes of the preceding ones.

The first step in the MVP workflow is the rebinning step, which processes the input data to reduce its volume, making subsequent steps more manageable. It uses parameters such as the input data path, rebinning flags and output location.

The calibration step follows the rebinning, employing calibration techniques to correct the data from any potential instrument bias based on given calibration parameters.

The imaging step is the last one and it takes the calibrated measurement set (or a collection of them) to generate an image cube with scientific value.

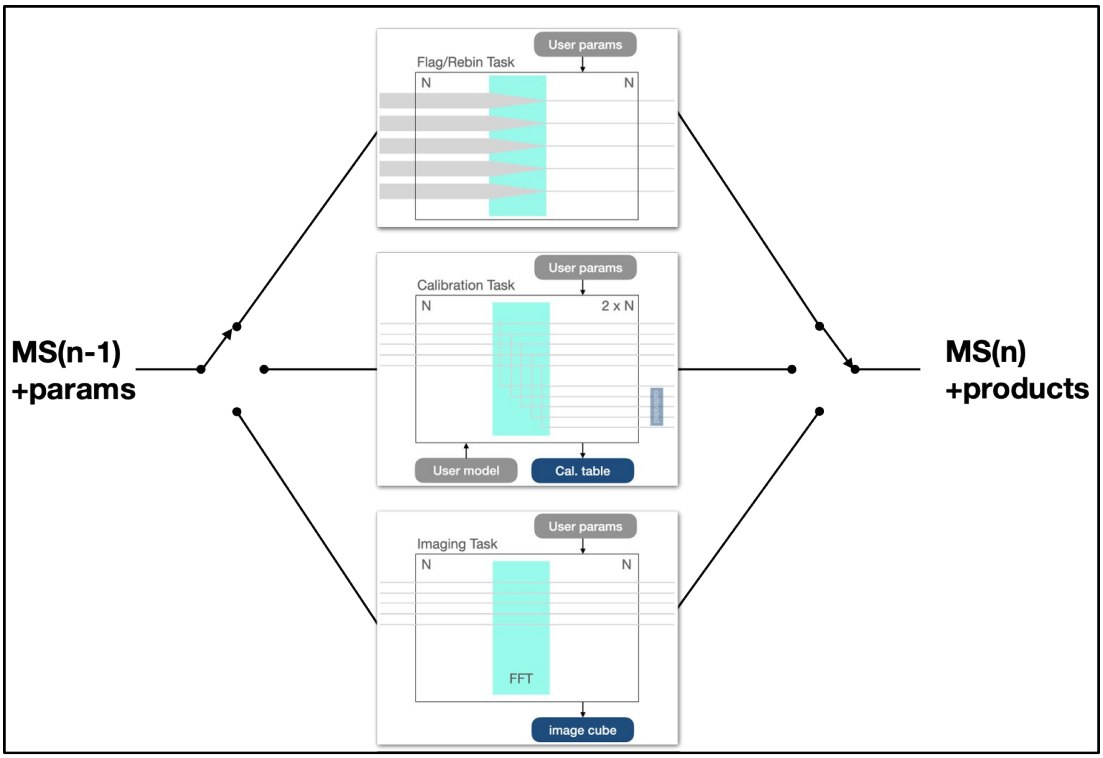



***Figure 6:*** *Principle of the sequential branching of task. At any given step, one of the tasks is performed. It takes inputs, produces outputs, and supplies them to the next task.*

### 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.

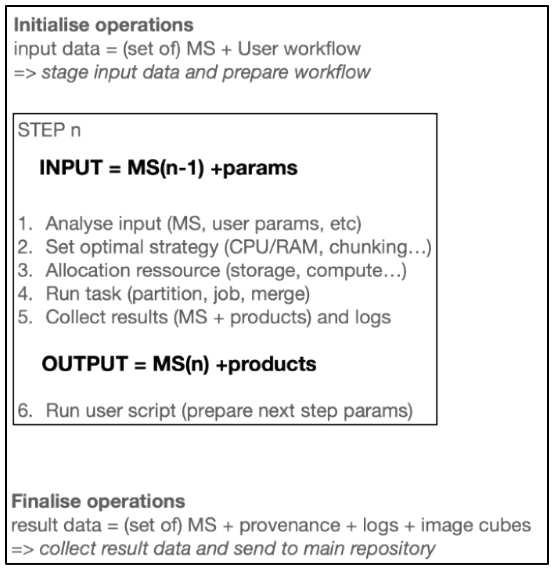



***Figure 7:*** *Generic workflow algorithm*

```
#Workflow definition

workflow = {
"RebinningStep": {
  "input_data_path": S3Path("/ayman-extract/partitions/partitions_61zip"),
  "parameters": {
    "flagrebin": {
        "steps": "[aoflag, avg, count]",
        ...
    }
  },
  "output": S3Path("/ayman-extract/extract-data/rebinning_out/"),
  }

}

#Workflow execution

rebinning_profilers = RebinningStep(
  input_data_path=S3Path(workflow["RebinningStep"]["input_data_path"]),
  parameters=workflow["RebinningStep"]["parameters"],
  output=workflow["RebinningStep"]["output"]
).run()
```

***Figure 8:*** *Workflow definition and execution*

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.

The parameters tell the script under the rebinning step what it should do. These parameters are passed to the script that is executing under the hood and are different for each step of the workflow.

Finally, we can pass the different parameters to the class and call the run method to enable the workflow to run. Each step should be defined separately, and all the steps take the same parameters.

Internally, this workflow execution leverages the EXTRACT platform to run these steps in the continuum. In this MVP, the computation is 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.

## 4.2. PER Use Case MVP

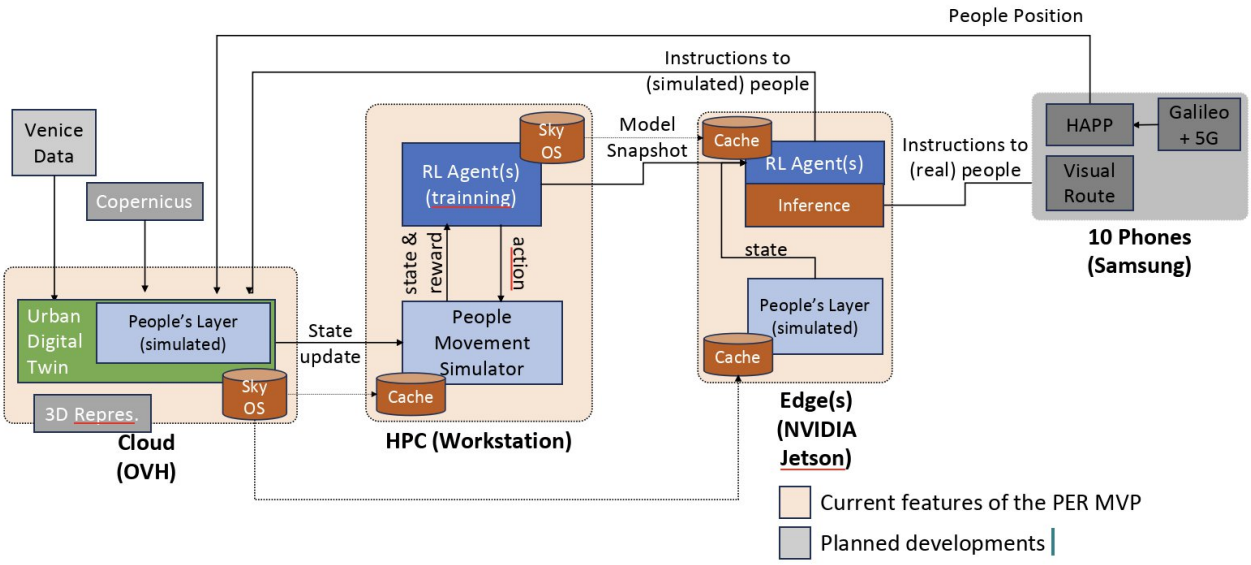



***Figure 9:*** *PER Use Case Architecture*

The PER Use Case MVP within the EXTRACT framework addresses emergency evacuation management in urban settings, with a focus on Venice's unique challenges due to its dense population, geographical configuration and urban structure. The MVP emphasizes core operational capabilities, technology integration, and strategic data use to improve evacuation efficiency and safety through AI.

The MVP, whose core componets have been highlighted in **Figure 9:**, implements the **semantic Urban Digital Twin (UDT**) for a meanigful modeling of Venice. It combines dynamic and historical data generated by the smart city, to enable the development of effective evacuation strategies by simulating various scenarios. **The MultiAgent Reinforcement Learning Model (MARL)** complements this by employing advanced machine learning techniques to refine evacuation routes based on evolving urban conditions, drawing on data from the UDT and additional sources.

A critical element of the MVP is the **simulation environment**, which allows for the thorough evaluation of evacuation strategies. It replicates individual movements using data and predictions from the UDT and MARL, facilitating the iterative improvement and validation of evacuation plans.

Additionally, the **Data Integration and Processing Framework** plays a vital role. It consolidates various data streams, including inputs from the Smart Control Room (SCR) and other sources, into a cohesive model. This supports efficient data handling and processing, essential for the timely generation of evacuation routes.

It should be noted that the MVP's current scope does not encompass the full integration and deployment of end-user applications such as the Evacuation Mobile App (EMA).

Instead, the focus remains on backend functionalities, including data management, simulation, and optimization of evacuation routes. This sets the groundwork for subsequent development phases involving user interface enhancements and real-time communication features.

The subsequent sections of this chapter provide a detailed overview of the MVP's operational capabilities, technological foundations, and data strategies. Starting with an assessment of the MVP's functional requirements and underlying data, the discussion progresses to an in-depth examination of implemented technologies, namely the UDT, MARL, the simulation environment, and the Data Integration and Processing Framework. This structure aims to lay a robust foundation for the eventual development of a comprehensive, personalized evacuation management system, addressing the specific challenges posed by Venice.

### 4.2.1. Overview of Requirements and Datasets for the PER MVP

#### Requirements

The minimum viable product approach involves prioritizing product requirements to the point that they deliver core functionality to deal with the general issue defined in D1.1. The exercise has been conducted using the Volere framework[1] with the early adopters (Venice City) that helped us in identifing the most critical requirements that need to be evaluated first.
These requirments have been classified in 3 categories: Basic funtional requirements, Performance requirements, Integration Requirements

| | |
|---|---|
| **Basic Functional Requirements**<br><br>These include the verification of the feasibility of the those features that characterise the PER application. | • **Urban Data Integration and Hazard Detection** The MVP is designed for collecting simulated data representing the characteristics of the data coming from the urban environment and integrate them properly with a semantic approach.<br><br>• **Personalized Evacuation Routes:** Leveraging simulated Venice's unique urban data, the system generates, personalized evacuation routes, accounting for individual and environmental variables using a MARL.<br><br>• **Simulator**: Simulating behaviours and emulating data coming from mobile devices (e.g. lat/long) of a given number of individuals moving on a selected area of the city. |
| **Performance Requirements** | • **Simulated Georeferencing Individuals:** Simulating a relevant number of positions of individuals in Venice is fundamental for a) early evaluation of the technology |

[1] https://www.volere.org/

| | |
|---|---|
| These include the verification of the existance of the potential for scalability of the technological appropchaces adopted | performance and b) testing the dataflow in the MEV. Moreover the simulation is needed for the the training phase of MARL based route calculation.<br>• **System Performance** The MVP is implemented taking into account the performance in order to be compliant with the experimental needs. |
| **Integration Requirements**<br><br>These include the verification that the designed dataflow among the components works as intended | • **Integration APIs:** Facilitating easy integration with the UDT, MARL and Simulator is the cornerstone of the MVP's functionality.<br>• **Data and Machine Learning Support:** Libraries for processing extensive datasets and supporting advanced machine learning are integral for informed, timely decision-making in emergency evacuation planning. |

## Datasets

The PER-MVP utilizes primary and supplementary datasets:

**Primary Data Sources:** Essential datasets include SCR include

- **Georeferenced position of individuals based on Galileo Satellite Data**: This dataset offers precise global navigation satellite system information, crucial for enhancing geolocation accuracy in real-time emergency scenarios, ensuring effective navigation and evacuation route optimization. In the MVP this data will be emulated by the Simulator.
- **Urban Generated Data:** These datasets include pedestrian flow metrics, water traffic observations, and detailed city maps.

These are vital for data integration and evacuation modeling implemented in the MVP.

**Supplementary Data Sources:** These include datasets like tidal levels and public transport schedules, and are useful to enrich the context/situation awareness for emergency response calculation (evacuation routes). This overview builds upon the detailed groundwork outlined in Deliverable D1.1, aiming to offer readers a clear understanding of the components underpinning the MVP development.

| | |
|---|---|
| **Source: Venice Smart Control Room Data** | • Pedestrian flow: computed using sensors capable of counting how many human beings are entering or exiting the monitored area. The overall number of sensors is 10, |

| | |
|---|---|
| | providing a number of more than 19.000 records in the Rialto zone. This datasets will be used to cross-validate the level of crowd density calculated by UTD in the MPV on the base of the data simulated by the Simulator.<br><br>• Water traffic: produced analyzing the video footage captured by constantly active cameras watching the Grand Canal and recognizing the direction as well as the type of ship that is passing through the zone under surveillance. This dataset can be used to validate the information related to the presence of the boat at the stop. This is crucial for the information reliability.<br><br>• Presence: the number of people present in the city computed from distinct mobile phone active SIM, updated every 15 minutes and aggregated in squares of 150m per edge. It provides more than 1 million of records; This datasets will be used to cross-validate the level of crowd density calculated by UTD in the MPV on the base of the data simulated by the Simulator.<br><br>• Spatial layers representing the city services useful for evacuation (e.g. places where is possible to find drinking water) and collecting points, obstacles (e.g. aquatic walkway), and so forth<br><br>• Pedestrian flow and water traffic datasets are provided in single JSON files. Presence dataset is provided in multiple CSV files. Spatial layers are provided in several archive files (shp, kmz) containing the layer geometry and metadata. |
| **Publicly Available Data** | The following are some additional useful datasets, that are publicly available, particularly valuable in the context of emergency management:<br><br>• The current level of the tide in the city of Venice, published in an official website as a JSON file and updated every about 5 minutes: https://dati.venezia.it/sites/default/files/dataset/opendata/livello.json<br><br>• The map with all the altimetry level with all the foot traffic level of the whole city: http://smu.insula.it/index.php@option=com_content&view=article&id=114&Itemid=81.html<br><br>• The maximum level of the tide in Venice for the main waterstops in the city, beyond which the waterstop cannot be used anymore; |

| | |
|---|---|
| | • The timetables of the public transport lines: https://avm.avmspa.it/en/content/orari-servizio-di-navigazione-0<br><br>• The map with all the waterstops position: https://www.google.com/maps/d/viewer?mid=1y-REIZ8dDZ_l3FCjHud8RF7buXPmpV0I&ll=45.441003066064637%2C12.342516890940448&z=14 |

### 4.2.2. Technological Framework and Scope of the PER MVP

This section delves into the core building blocks foundational to the MVP for the PER Use Case, specifically:

1. The Urban Digital Twin (UTD)
2. The MARL
3. The Simulator
4. The Data Integration and Processing Framework.

Each component of the MVP is described focusing its its impact on the MVP's operational objectives.

**1.The UDT:** forms the MVP's foundational block, presenting a dynamic virtual representation of the selected are of Venice. It acts as data recipient and model-based sense-making. By incorporating data from the SCR and other sources, the UDT collect and represents the effects of the urban responses to the emergencies accurately, crucial for the implementation and continuous adjustment of changing conditions of the evacuation mission. It integrates information on street adjacency, potential hazards, available bandwidth, and people's positions, laying the groundwork for informed scenario planning and decision-making.

**2.The MARL** is vital for the strategic advancement of the MVP, employing data from the UDT to compute and refine evacuation routes taking into account varied urban conditions. By responding to evolving scenarios, such as changes in crowd dynamics and environmental hazards, the MARL's adaptive algorithms are key in optimizing evacuation routes, ensuring they are effective and timely.

**3.The Simulator** is an indispensable component, working closely with the UDT and MARL to provide a realistic environment for simulating individual and crowd movements during evacuations. This tool is crucial for validating the efficacy of proposed strategies, allowing for their refinement in a risk-free setting. Particularly during the model training phase, the Simulator emulates step-by-step people movements based on MARL's recommendations, testing the practicality and efficiency of evacuation actions.

**4.The Data Integration and Processing Framework** consolidates diverse data streams into a unified, actionable model, essential for enabling swift and effective decision-making. By facilitating the seamless amalgamation of information, this

framework underpins the optimization of evacuation paths, ensuring emergency responses are both prompt and informed.

While the UDT, MARL, Simulator, and Data Integration and Processing Framework currently constitute the technological backbone of the MVP, they are integral to its immediate operations and are targeted for iterative enhancements to improve the system's efficiency and adaptability.

To better illustrate the collaborative interaction among these technologies, a graphical representation that can simplify this interaction is inserted here. This visual aid simplifies the complex data flows and interactions between the UDT, MARL, Simulator, and Data Integration and Processing Framework, offering a clear, intuitive understanding of how these components work together to optimize emergency response strategies.

Looking ahead, the project's trajectory includes expanding the MVP's capabilities to address emergency situations more effectively.

### 4.2.3. Role of the Urban Digital Twin in the PER MVP framework

The MVP utilizes the UDT as a foundational element for integrating heterogeneous urban data and continuously compute the state of the city to take informed decisions in emergency scenarios. The UDT integrates dynamic and static urban data, facilitating the calculation of dynamic evacuation routes and improving situational awareness—key aspects of the MVP's operational efficacy.

Within the MVP framework, the UDT functions beyond a traditional modeling tool; it operates as an active asset influencing the formulation and deployment of evacuation strategies. By assimilating real data from Venice along with simulated data, the UDT enables practical insights from complex urban dynamics analysis, ensuring responsive measures to emergencies.

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 imply the definition of an ontology representing a semantic model of the city managed through the Virtuoso triple store[2]. This integration employs the W3C-OGC SOSA (Sensor, Observation, Sample, and Actuator) ontology[3], 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 of elaboration. This file subsequently informs the Reinforcement Learning component developed by BSC, facilitating advanced urban dynamics analysis and simulation. This approach enhances the MVP's capability to deliver informed and timely responses under varying urban emergency conditions. 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

[2] https://virtuoso.openlinksw.com/

[3] /www.w3.org/TR/vocab-ssn/

necessary to update the status again), the rate 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.

**Offline Tool Development (LRI Component)**

Overview

The UDT involves the development of a specialized Python script that plays a crucial role in transforming urban data into a format suitable for semantic processing and integration into the project's knowledge base. This script processes OpenStreetMap (OSM) data to generate a comprehensive representation of urban elements like roads and buildings.

Script Functionality

Data Extraction and Processing: The graphGenerator.py script is designed to extract geographical and structural data from OpenStreetMap. It focuses on retrieving detailed information about various urban elements, particularly the layouts of road networks. Figure 10 shows a code snippet that generates the GraphML file, and Figure 11 presents a plot of the generated graph.

```
import osmnx as ox
from shapely.geometry import Polygon

cords = (
    (12.3047, 45.4455), (12.2995, 45.4389),
    (12.3052, 45.4368), (12.3084, 45.4411),
    (12.3140, 45.4418), (12.3061, 45.4463),
    (12.3047, 45.4455))

polygon = Polygon(cords)

G = ox.graph_from_polygon(polygon, network_type='all', retain_all=True, simplify=True, truncate_by_edge=True)
M = ox.utils_graph.get_undirected(G)

ox.save_graphml(M, filepath="./road_graph.graphml")
```

***Figure 10**: Code Snippet that generates the GraphML*

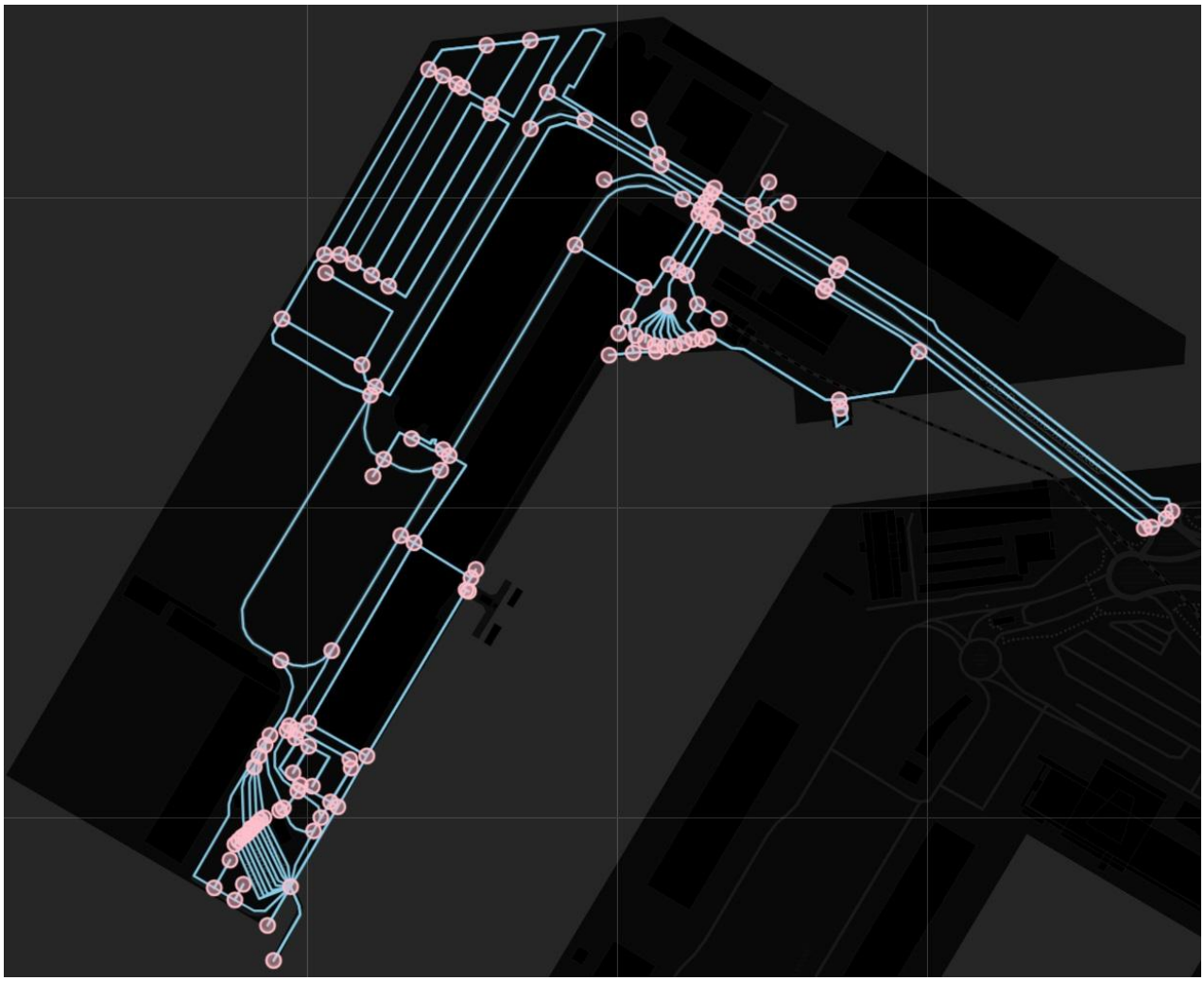


***Figure 11**: Generated Graph*

Ontology Mapping and RDF Generation: The osm_road2ttl.py script takes the GraphML file and maps the data onto a specially designed road ontology. This process involves converting the graph data into RDF (Resource Description Framework) triplets, a standard format for encoding metadata and other information on the web. This conversion is vital for semantic processing and integration of this data into larger knowledge bases.

**Ontology Description**

A customized and simplified version of the km4city ontology, integrated with elements from the OTN (Open Transport Network) ontology for road graph descriptions and the SOSA ontology for mobile device and traffic observations. The ontology is structured into four macro-classes: Road, Features, Observations and Events, each catering to different aspects of urban dynamics and data representation.

Road Class: This class is based on the OTN ontology. It includes the following key components:

- **EXT.Road_Element**: This is a subclass of OTN.Road_Element.
- **Relationships with EXT.Node**: Each Road_Element is connected to EXT.Node through two object properties, OTN.starts_at and OTN.ends_at.

- **Relationships with EXT.Road**: A EXT.Road_Element is linked to EXT.Road through two inverse object properties, EXT.IsPartOfRod and EXT.ContainsRoadElement.
- **GEO.Geometry Class**: This class is related to both the Road_Element class and the Node class via the object property GEO.hasGeometry.

This structure allows for a detailed representation of road elements, their start and end points, and their inclusion within larger road networks. The use of geometry information further enhances the spatial representation of these elements. The development is ongoing, with more information being added to the ontology.

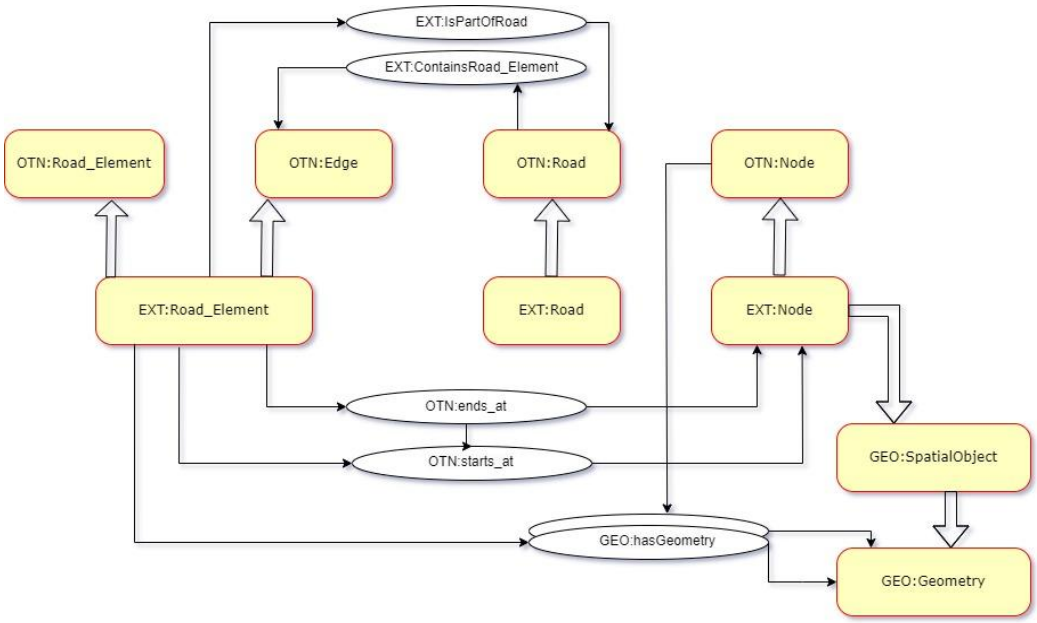



***Figure 12**: UML Diagram 1*

Observation class: This class is pivotal in representing dynamic and real-world data within the ontology. This class includes two primary subclasses: EXT.TrafficObservation and EXT.DeviceObservation, both of which are derived from the SOSA.Observation class.

- **EXT.DeviceObservation**: This class is intricately linked with EXT.MobileDevice, a subclass of SOSA.Sensor, through two inverse object properties: SOSA.madeBySensor and SOSA.madeObservation. This connection illustrates the relationship between the observation and the device (sensor) that made the observation.
- **Geospatial Representation**: EXT.DeviceObservation is also associated with GEO.Geometry through the object property EXT.hasPosition. In this context, GEO.Geometry represents the geographical location of the observation, typically as a geo-coordinate point.
- **EXT.TrafficObservation**: This class, along with EXT.DeviceObservation, is linked to the TIME.Instant class via the object property EXT.ObservationTime. This connection indicates the specific time at which the observation was made, thereby providing a temporal context to the data.

Overall, the Observations class in the ontology serves to encapsulate sensor-based data and traffic-related information, grounding them both in spatial and temporal

dimensions. This structure is essential for analyzing and understanding the dynamics of urban environments in real-time.

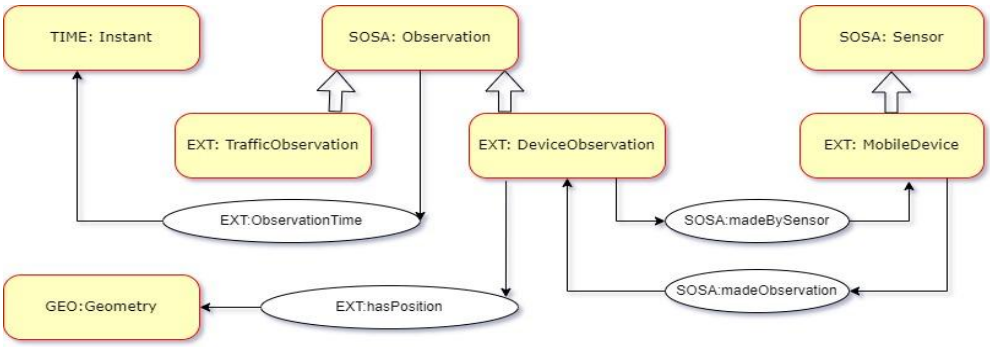



***Figure 13**:UML Diagram 2*

Features Class: The Features macroclass, as a part of our ontology, is structured under the GEO.Feature superclass, a key component of the GeoSPARQL ontology. This class is designed to encapsulate various urban features and is detailed as follows:

- Subclasses:

The Features class includes several subclasses representing different urban elements. These include:

  - **GreenField**: Represents green spaces or undeveloped land within the city.
  - **Building**: Denotes structures or buildings.
  - **Barrier**: Identifies physical barriers like walls or fences.
  - **Bridge**: Signifies bridge structures within the city.
  - **Pier**: Represents piers or similar waterfront structures.

These subclasses provide a comprehensive representation of diverse urban features, contributing to a detailed urban model.

- Relationship with Geometry: Each feature within the Features class is linked to geometric data through the GEO.hasGeometry object property.
- This relationship allows each feature to be associated with a specific geolocation, which can be represented as either a Point or a Polygon.

The incorporation of the Features class into the ontology enhances the spatial and physical understanding of the urban environment, offering a multifaceted view of the city's layout. This detailed representation is crucial for accurate urban modeling and analysis. The ongoing development of the ontology aims to further enrich this class with more features and refined relationships.

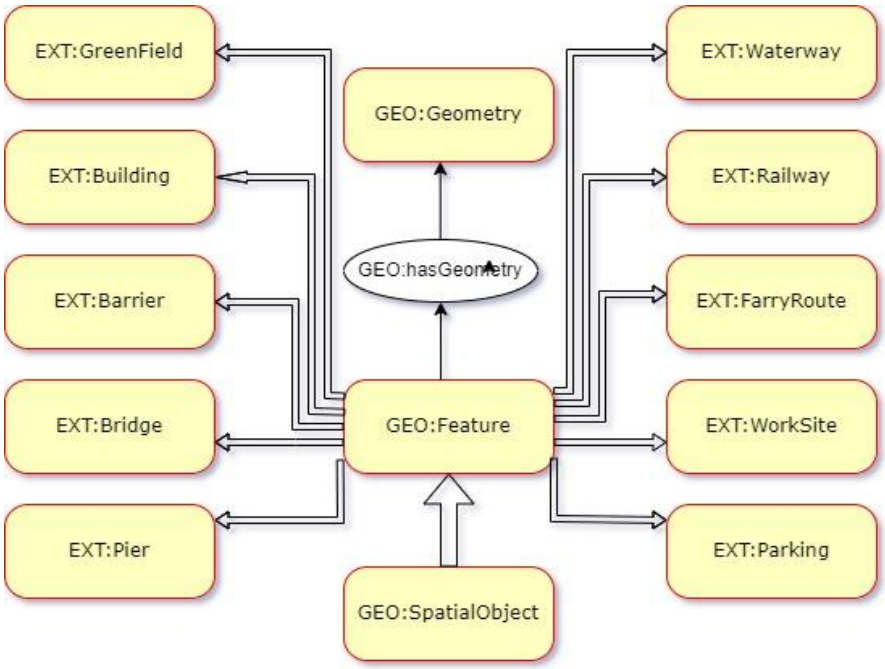



***Figure 14**: UML Diagram 3*

Events Class: The Events class within our ontology plays a crucial role in representing various occurrences within the urban environment. This class is structured as follows:
The Events class is divided into two main subclasses, which are part of the schema:Event ontology:

- Subclasses:
  - **EXT.HumanMadeEvent**: This subclass encompasses events that are caused by human activities. Examples include:

  - **Fire**: Denotes occurrences of fires, whether accidental or intentional.
    Note: In this section can be inlcuded other events as ChemicalLeak (that represents incidents involving the accidental release of hazardous chemicals) or TerroristAttack (that can represent an acts of terrorism or similar violent events).

  - **EXT.NaturalEvent**: This subclass covers events that are natural in origin. Examples may includeFlood (that Represents flooding events, which can be significant in a city like Venice), Earthquake (that denotes seismic events or earthquakes), Flash floodings, etc.

In the contex of the EXTRACT project, we focus the Fire event.

- Temporal Relationships:
  Events in both subclasses are linked to temporal data using properties from the TIME ontology:
  - **time.hasBeginning**: This property indicates the start time of an event.
  - **time.hasEnd**: This property specifies the end time of an event.

- Spatial Relationships:
  Events are also associated with specific locations using the schema:Place ontology:
  - **schema.Location**: This object property links an event to its geographical location, providing spatial context.

The Events class is particularly significant in the context of urban dynamics, capturing both natural and human-made occurrences that impact city life. This class is currently under development, with ongoing efforts to incorporate more detailed information about events specific to Venice.

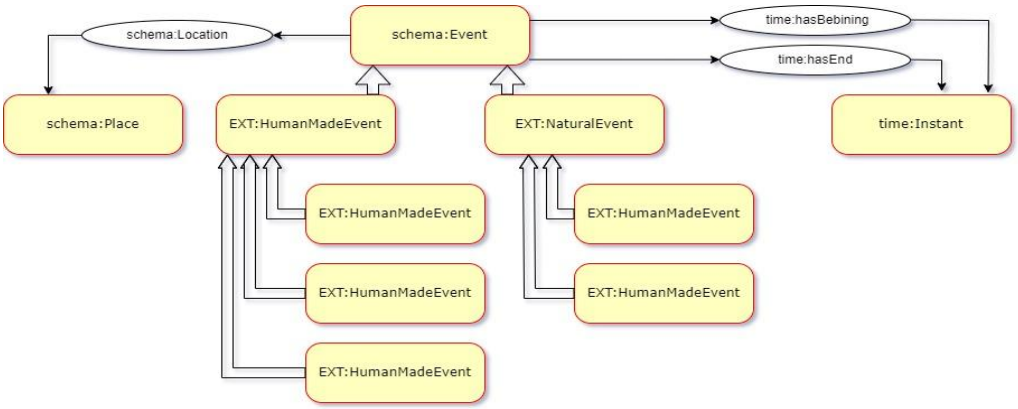



***Figure 15:*** *UML Diagram 4*

**Importing OpenStreetMap (OSM) data into the knowledge base**

The script "osm_road2ttl.py", is designed for importing OSM data into the knowledge base.

The script includes functions for loading GraphML files and ontologies, creating geometries from edge data in the graph, generating unique identifiers for road elements based on OSM IDs, and estimating road widths based on various attributes.

Key functionalities of this script include:

- **Loading GraphML files**: This is likely used to import road network data from OpenStreetMap in the GraphML format.
- **Creating geometries**: The script can create geometries such as LineStrings from edge data and Points for nodes which are essential for representing roads and paths in a geographical information system.
- **Generating unique IDs**: The script includes a method to generate unique identifiers for road elements, roads and nodes which is crucial for organizing and referencing specific road segments within the knowledge base.
- **Estimating road_element widths**: In cases where road width information is not present in the OSM data for a particular area, estimating the width based on available information becomes necessary. This process typically involves using factors such as the number of lanes, road direction, or road type. In your

project, this estimation is implemented in the osm_road2ttl.py script. The width of the road segment is estimated using the *estimate_width(data)* function. This estimation takes into account factors like the number of lanes and road type. The estimated width is then used, along with the length of the road segment *calculated by length_cal(data)*, to estimate the area of the road segment. The area is calculated as the product of the estimated width and the road length. This approach allows for a more comprehensive representation of road elements in the OSM data, even when explicit width information is not available

OSM Nodes Mapping:

The transformation process of node data from GraphML format to RDF (Resource Description Framework) is outlined below. This conversion is a crucial step in mapping OSM data to the ontology within the MVP framework.

Extraction of Node Data from GraphML: Initially, the GraphML file contains node data in a structured XML format. An example of a node entry in GraphML format is as follows:

```
<node id="9196476007">
  <data key="d4">45.4418253</data>
  <data key="d5">12.3075776</data>
  <data key="d6">3</data>
</node>
```

In this example, the node is identified by *id="9196476007"*, and it includes data elements like *latitude (45.4418253)* and *longitude (12.3075776)*.

Mapping nodes to RDF Format: The node data from GraphML is then transformed into RDF triples. Each node in GraphML is represented as an instance of otn:Node and ext:Node in RDF, and it is associated with its geometrical data. For example, the RDF representation of the above node is:

```
<https://www.extract-project.eu/ontology#Node/9196476007> a otn:Node,
        ext:Node ;
    geo:hasGeometry <http://www.opengis.net/ont/geosparql#Geometry/9196476007> .
```

```
<http://www.opengis.net/ont/geosparql#Geometry/9196476007> a geo:Geometry ;
    geo:asWKT "POINT (12.3075776 45.4418253)"^^geo:wktLiteral .
```

In this example, the node is identified by a unique URI constructed from its GraphML id. The geo:hasGeometry property links the node to its geometrical representation (a point with latitude and longitude).

Extraction of Edge Data from GraphML: The representation of an edge in a GraphML file, especially in the context of road networks derived from OSM, can vary based on the complexity and structure of the underlying road elements. Here are the key points regarding how edges are defined and identified:

- **Basic Edge Definition**: In a GraphML file, an edge is primarily defined by its 'source' and 'target' nodes. These nodes represent the start and end points of the edge, effectively capturing a road segment's direction and connectivity in the network.

- **OSM Way IDs for Edges**: Each edge corresponds to a way in OSM. A way in OSM is a sequence of nodes that define polyline features such as streets, paths, or boundaries. In the GraphML file, each edge is associated with an OSM way ID, which is a unique identifier assigned to that specific way in the OSM database.

- **Edges with Multiple OSM Ways**:
  - In some cases, an edge in the GraphML file may be composed of multiple OSM ways. This situation occurs when a road segment is represented by a sequence of smaller segments in OSM, each having its own way ID.
  - When multiple OSM ways constitute a single edge, the edge's ID in the GraphML file is formed by concatenating the IDs of the individual ways, typically separated by underscores. For example, if an edge is made up of two OSM ways with IDs 'way_osm_id1' and 'way_osm_id2', the edge's ID would be 'way_osm_id1_way_osm_id2'.

- **Single OSM Way for Multiple Edges**:
  - Conversely, different edges in the GraphML file can be part of the same OSM way. This scenario is common when a longer OSM way is segmented into smaller parts, each represented as a separate edge in the graph.
  - In this case, the individual edges will have IDs that include the common OSM way ID followed by an index number to distinguish between the segments. For instance, if a single OSM way 'way_osm_id' is divided into three segments in the GraphML file, the edges might be identified as 'way_osm_id_1', 'way_osm_id_2', and 'way_osm_id_3'.

Initially, the GraphML file contains edge data in a structured XML format. An example of 3 edge entry part of the same osm way id in GraphML format is as follows:

```
<edge source="2123929755" target="5853655237" id="0">
  <data key="d8">450261871</data>
  <data key="d9">footway</data>
  <data key="d11">False</data>
  <data key="d12">True</data>
  <data key="d13">4.083</data>
  <data key="d14">5853655237</data>
  <data key="d15">2123929755</data>
  <data key="d16">LINESTRING (12.3092629 45.4422112, 12.3092898 45.4422427)</data>
</edge>
```

```
<edge source="2123929755" target="2123929756" id="0">
  <data key="d8">450261871</data>
  <data key="d9">footway</data>
  <data key="d11">False</data>
  <data key="d12">False</data>
  <data key="d13">12.201</data>
  <data key="d14">2123929756</data>
  <data key="d15">2123929755</data>
  <data key="d16">LINESTRING (12.3093707 45.4423366, 12.3092898 45.4422427)</data>
</edge>
<edge source="2123929756" target="5853655236" id="0">
  <data key="d8">450261871</data>
  <data key="d9">footway</data>
  <data key="d11">False</data>
  <data key="d12">False</data>
  <data key="d13">5.071</data>
  <data key="d14">5853655236</data>
  <data key="d15">2123929756</data>
  <data key="d16">LINESTRING (12.3094044 45.4423756, 12.3093707 45.4423366)</data>
</edge>
```

Mapping edges to RDF Format: The edge data from GraphML is then transformed into RDF triples. Each edge in GraphML is represented as an instance of otn:Road_Element and ext:Road_Element in RDF, and it is associated with its geometrical data. For example, the RDF representation of the above edges with osm way id 450261871 is:

```
<http://www.pms.ifi.uni-muenchen.de/OTN#Road/450261871> a otn:Road ;
    otn:contains <http://www.pms.ifi.uni-muenchen.de/OTN#Road_Element/450261871_1>,
        <http://www.pms.ifi.uni-muenchen.de/OTN#Road_Element/450261871_2>,
        <http://www.pms.ifi.uni-muenchen.de/OTN#Road_Element/450261871_3> .
<http://www.pms.ifi.uni-muenchen.de/OTN#Road_Element/450261871_1> a otn:Road_Element ;
    geo:hasGeometry <http://www.opengis.net/ont/geosparql#Geometry/450261871_1> ;
    otn:ends_at <https://www.extract-project.eu/ontology#Node/5853655237> ;
    otn:starts_at <https://www.extract-project.eu/ontology#Node/2123929755> ;
    ext:area "8"^^xsd:float ;
    ext:estimatedWidth "2"^^xsd:float ;
    ext:length "4.083"^^xsd:float .
<http://www.pms.ifi.uni-muenchen.de/OTN#Road_Element/450261871_2> a otn:Road_Element ;
    geo:hasGeometry <http://www.opengis.net/ont/geosparql#Geometry/450261871_2> ;
    otn:ends_at <https://www.extract-project.eu/ontology#Node/2123929756> ;
    otn:starts_at <https://www.extract-project.eu/ontology#Node/2123929755> ;
    ext:area "24"^^xsd:float ;
    ext:estimatedWidth "2"^^xsd:float ;
    ext:length "12.201"^^xsd:float .
<http://www.pms.ifi.uni-muenchen.de/OTN#Road_Element/450261871_3> a otn:Road_Element ;
    geo:hasGeometry <http://www.opengis.net/ont/geosparql#Geometry/450261871_3> ;
    otn:ends_at <https://www.extract-project.eu/ontology#Node/5853655236> ;
    otn:starts_at <https://www.extract-project.eu/ontology#Node/2123929756> ;
    ext:area "10"^^xsd:float ;
    ext:estimatedWidth "2"^^xsd:float ;
    ext:length "5.071"^^xsd:float .
```

The process of data ingestion from OSM using OSMnx and the transformation using the Python script developed by LRI generates an RDF file representing the road graph. This RDF file encapsulates various details about the road network, including nodes, edges, and their spatial properties. Once generated, this RDF file is uploaded to a Virtuoso triplestore instance, which is running in a Docker container. This allows for efficient storage and querying of spatial data, serving as a foundation for further analysis and integration with real-time city data.

**Importing simulated dynamic data into the Knowledge Base**

Mapping Real-Time Data to the Observation Ontology involves integrating dynamic, real-time data from InfluxDB into the existing ontology. This process primarily includes the positioning data of real and simulated individuals in Venice. The extracted data are linked to specific ontology classes such as EXT.TrafficObservation and EXT.DeviceObservation, both subclasses of the SOSA.Observation class. These classes represent the movements and interactions of people within the urban environment. This data integration facilitates the real-time analysis and representation of urban dynamics, contributing significantly to the comprehensive monitoring and understanding of the city's state.

The workflow for incorporating real-time data into the urban monitoring system can be described as follows:

- **Dynamic Data Collection**: Data is primarily sourced from the time series DB as InfluxDB[4], which includes time stamped positional data of simulated individuals . Additional data sources in development include inputs from the city of Venice.
- **Mapping to Observation Ontology**: The dynamic data collected is used to populate and update the device observation ontology. This includes storing data about individual device locations and times.
- **Traffic Observation Data Generation**: Utilizing the dynamic positional data, key metrics such as the average speed and density of people at specific nodes and roads are calculated. This information is then integrated into the traffic observation ontology, with time-stamped records for ongoing monitoring.

[4] https://www.influxdata.com/

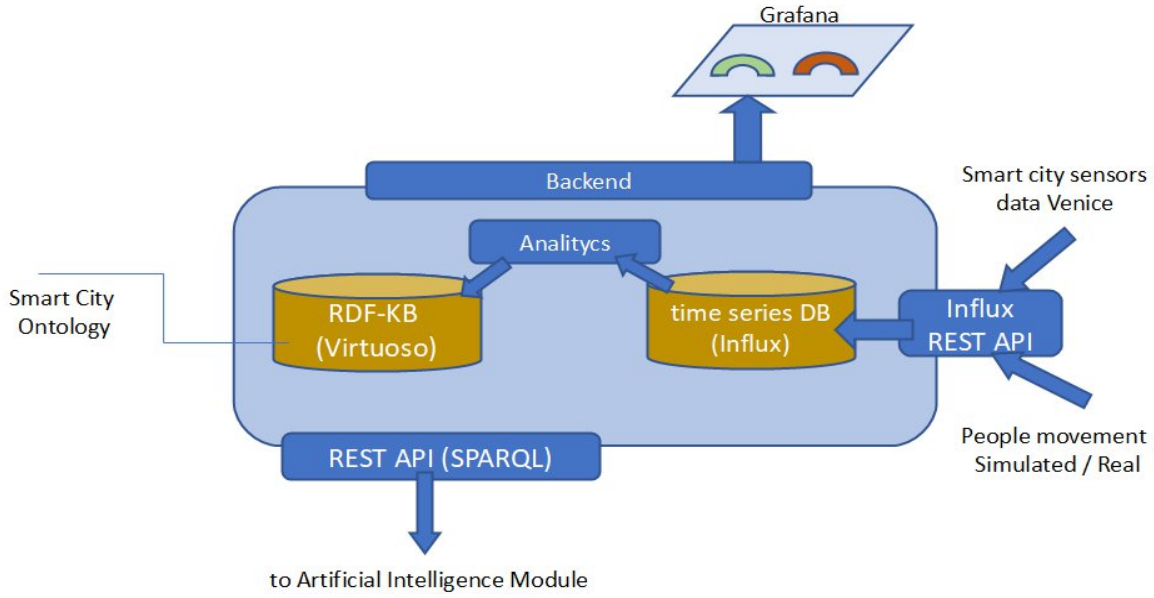



***Figure 16**: UDT MVP architecture*

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.

### 4.2.4. Emergency Notification and Urban Digital Twin Update

**Notification of Emergency Events**

The initiation of the PER Use-Case MVP'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. Activation of Reinforcement Learning for Evacuation Optimization

Following the emergency notification and the comprehensive update of the Urban Digital Twin, the next critical step involves the activation of the Reinforcement Learning (RL) for evacuation optimization. This phase is pivotal in translating the updated urban data into actionable evacuation strategies, ensuring that residents and visitors in Venice are guided to safety in the most efficient manner possible during emergencies.

**Introduction to Reinforcement Learning**

The narrow streets and canals of Venice make the city unique, and pose significant challenges for evacuation. Traditional centralized routing systems might struggle to adapt in real-time to the evolving conditions, such as crowd densities, rising waters or blocked pathways.

To that end, MARL is a perfect fit to address these conditions. MARL can address the complexity of Venice’s topology by training multiple agents that learn and adapt to this evolving environment. Each agent in MARL, represents an individual or a group within the crowd, making decisions on their best route to safety.

These agents learn from the environment through trial and error, and might even develop strategies that consider both personal and overall safety in the evacuation.

Multi-agent reinforcement learning is particularly promising for this project, as it allows for the collaborative exploration of optimal paths to safety for multiple individuals or groups within the simulation. By coordinating the actions of multiple agents, such

algorithms can efficiently navigate complex environments, considering the interactions and dependencies among different entities to achieve optimal outcomes. This collaborative approach can enhance the overall effectiveness and robustness of the system in finding safe paths for individuals or groups amidst dynamic and challenging scenarios.

**Initiation of Reinforcement Learning**

The initiation of RL is a procedural response to the updated scenario as presented by the Urban Digital Twin. This step marks the commencement of a sophisticated computational process where the RL system begins to analyze the current urban environment, integrating new data to simulate various evacuation scenarios. The RL system's objective is to understand the dynamics of the evolving situation and to identify the optimal paths for evacuation that consider numerous variables, including crowd density, obstacle locations, and individual mobility profiles.

**DDPG**

Currently, the reinforcement learning algorithm employed in the PER use case is Deep Deterministic Policy Gradient (DDPG). This choice is motivated by the algorithm's capability to handle continuous 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 continuous action spaces; essentially, they could only handle the movement of a single individual at a time. Conversely, DDPG has demonstrated efficacy across diverse tests encompassing varying map sizes and population densities. Moving forward, additional single-agent algorithms like TD3 and Soft Actor Critic will undergo implementation and testing, alongside multi-agent algorithms like Q-Mix.

MARL is currently undergoing development and is anticipated to substitute DDPG, particularly in scenarios necessitating collaboration among individuals. In such cases, MARL emerges as a superior choice due to its ability to model interactions between multiple agents effectively. In practical terms, each agent within MARL can represent either an individual or, more probably, a group of individuals expected to act collectively. This collective representation proves advantageous, as assigning a distinct agent for each person would incur significant computational overhead, rendering it impractical. Each group would have similar physical characteristics.

**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.

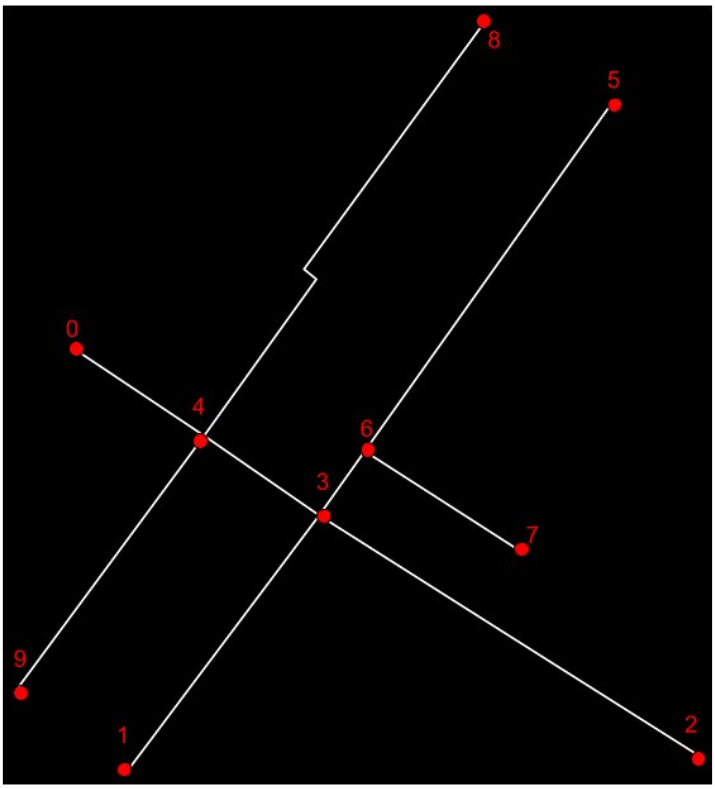
The BW channel effectively communicates the capacity of a street, indicating the number of individuals it can accommodate at any given moment. For instance, if a street initially had a capacity of 100 persons and 20 people are currently traversing it, the BW value would be adjusted to 80. It also serves 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.



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 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 action is represented by a 1-channel array with the same dimensions as the state. This array signifies the individuals that the RL agent aims to move:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, **4**, **6**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

The action has value 4 in the position 11, position 11 in the conversion array contains tuple (0, 4), representing the edge that goes from node 0 to 4, so the four persons found in node 0 will be moved to node 4. Also 6 people are moved from node 1 to node 3 in index 12. The state will be modified:

[**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]

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]

**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. For example, potential rewards include:

- Count of individuals located within the simulation:

  *reward = sum(next_state[:, 1]) * -1*

  Individuals' positions are tracked in the second channel of the simulation data.
- Total time taken to rescue all individuals:

*reward = current_time_step * -1*

The negative sign preceding the reward value signifies our objective of maximizing the reward, prompting the agent to minimize the reward value.

**Neural Network architecture**

There are two neural networks, actor and critic.

- Actor: Also referred to as the policy, the actor receives the state as input, where the state has dimensions (*n_nodes* + *m_edges*, 2), with 2 denoting the bandwidth channel and the people channel. The neural network's output is the corresponding action, characterized by dimensions (n_nodes + m_edges).

  *Actor(*
  *(fc1): Linear(in_features=(n_nodes + m_edges, 2), out_features=hidden_size)*
  *(fc2): Linear(in_features=hidden_size, out_features=hidden_size)*
  *(out): Linear(in_features=hidden_size, out_features=(n_nodes + m_edges))*
  *)*
- Critic: The critic accepts both a state and an action as input, with dimensions ((n_nodes + m_edges, 2) + (n_nodes + m_edges)). It produces a singular value representing the Q value, indicating the expected return for the given state-action pair.

  *Critic(*
  *(fc1): Linear(in_features=((n_nodes + m_edges, 2) + (n_nodes + m_edges)), out_features=hidden_size)*
  *(fc2): Linear(in_features=hidden_size, out_features=hidden_size)*
  *(out): Linear(in_features=hidden_size, out_features=1)*
  *)*

As the size of the state that needs processing increases, the complexity required for the neural network also grows. In simpler terms, a larger map necessitates a correspondingly larger hidden size for the models. Usually, a hidden size between 62 and 256 is used.

**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 MVP aims to deliver a highly adaptive and responsive system that can manage the complexities of emergency evacuation in the unique urban environment of Venice.

### 4.2.6. People Movement Simulator

As previously described, the correct generation of evacuation routes by MARL goes through a trial/error process, which is essential for managing the movement of individual users or groups of people in the streets of Venice.

Since it is fundamentally impossible to implement this trial/error process in the real world in a short time, to be able to train the RL model an ad hoc motion simulator was created, capable of receiving the data coming from the UDT and applying the commands generated by MARL.

The simulator allows to test the actions proposed by the RL module "in the field", making users move digitally in the most natural and realistic way possible. A true-to-life simulator offers a safe and controlled platform for testing learning strategies in different situations, even those not reproducible in real life (e.g. after an explosion or a fire). Furthermore, an accurate simulator can incorporate a wide range of environmental and behavioral variables, allowing learning algorithms to address a variety of real-world and complex challenges.

**Simulator initialization**

The simulator interfaces with two other components of Extract: on the one hand it obtains data on the position of users and their characteristics directly from the UDT, and on the other it works closely with the MARL, with which it interacts in a loop throughout the model training phase, simulating the actual movements of people.

The simulation is carried out using the same information base used by MARL, to guarantee uniformity in the management of movements from a geographical point of view: the topology on which the entire processing is based is in fact that coming from OSM, appropriately processed and managed, in a to be able to provide the scenario within which users can "move" during evacuation scenarios.

The data obtained from OSM, in addition to allowing a geographical location of the users in the city of Venice, makes it possible to manage the possible routes available to each individual, i.e. the roads, bridges, walkways, etc., present in the territory. Two main entities are extracted from the city topography, which are fundamental for the purposes of the simulation, and these are also shared with the MARL for uniformity in processing:

- nodes: points of intersection between two or more roads
- edges: the roads that connect two separate nodes

Obviously the "road" entity will not only be the edge that connects two nodes (which we can understand as a straight line that does not consider natural or artificial obstacles between the two nodes it connects), but rather the "real path" of this, made up of curves, turns or temporary obstacles (i.e. roadworks) and, in the case of Venice in particular, passages necessary to get around obstacles that otherwise cannot be crossed by a user, such as the city's canals.

While in the final version of PER the position and characteristics of the user will be provided by the sensors located in the city to the UDT, in this phase of training of the model, the initial positions of the users are generated by the simulator to allow different and customizable configurations for the training. Each user has personal characteristics, linked to his movement on the topography of Venice, which will

necessarily have to be taken into consideration during the simulation phase of their movements, to ensure that this is as correct and faithful to reality as possible

These personal characteristics are linked to position and movement, such as:

- geographical position: expressed in latitude and longitude, necessary to identify the user's position when calculating his best evacuation route

- heading: where the user is going, information which, through the merging of this data with those coming from OSM, allows us to be able to say "towards which node the user is proceeding"

- speed: what is the speed at which the user is (on average) moving, necessary to be able to correctly simulate his movements following the reception of the action to be applied to him

Of these three characteristics, the most fundamental for simulation purposes is undoubtedly the geographical position: without this it is in fact impossible to proceed with the user's movement. The other two, heading and speed, can be easily assumed: when absent we can assign the user a speed defined as "standard" (i.e. approx. 1.4 m/s), and consider that he is directed (heading) towards the node closest to him.

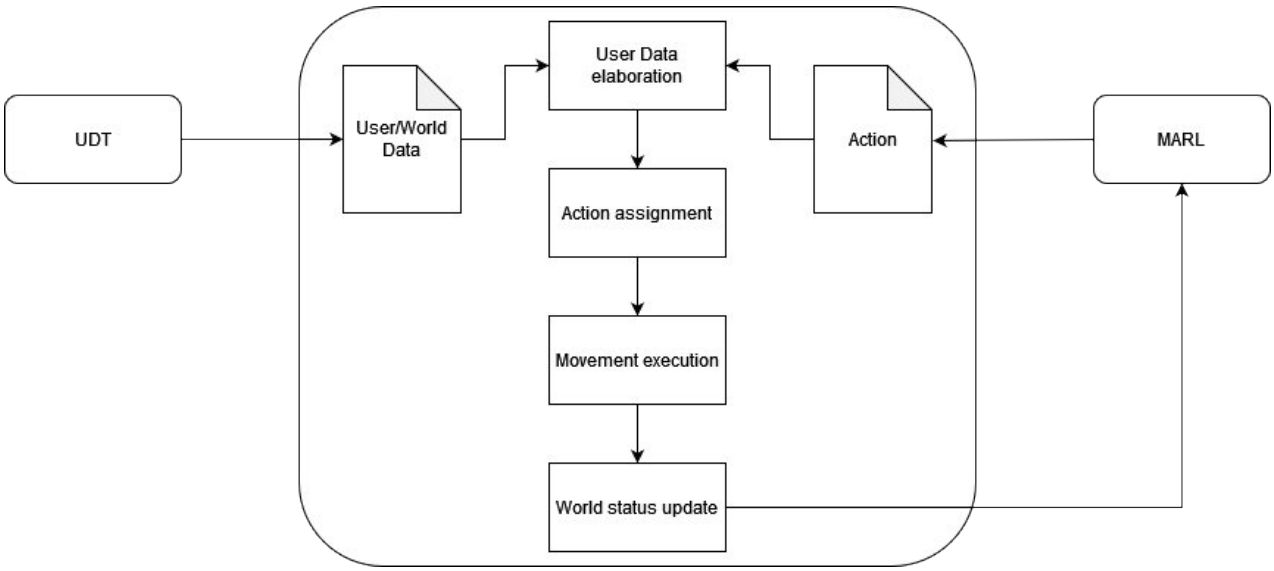
Each individual user will then be placed on the map in the geographical position expressed in latitude/longitude (we recall that in the final versions these data will be provided by the UDT), however, since this information is linked to the topology of the streets of Venice, we can also associate with each coordinate the OSM node or edge on which the person is located, thus standardizing the geographical data available to the simulator with that necessary for the processing of the PER by MARL.

Furthermore, through the UDT it is expected to also receive data relating to the state of danger and safety of the various points of the city. In fact, it is possible to imagine that, during the state of emergency, whether this is identifiable as fire, rising water or other situations, the epicenters or in any case the "danger zones" are identifiable on the map, in the same way as the areas designated as "gathering zones" to be able to gather people together as events unfold. We can identify them as:

- dangerous nodes: those nodes that are located at the epicenter of the dangers or in any case that are to be considered "to be avoided" during the escape route

- safe nodes: those parts of the map indicated as gathering areas, in which to gather users, and which can be considered as the preferable destinations for an evacuation route

**Simulation flow**




Once the simulator has been initialized, it is ready to receive the set of actions to be applied to users by the MARL and apply it to the data relating to the environment and people giving rise to the recursive trial/error process, which leads to the training of the model dedicated to PER.

Within this process we can identify three different main phases:

**- Action assignment**

As seen previously, from the MARL the simulator obtains an array containing the action to be applied to the users, expressed as the desired number of people in each node (intersection) of the OSM topology.

By combining this information with that already available to the simulator, which as explained is able to convert the position of the users from latitude/longitude to OSM node/edge, the algorithm is able to correctly assign to each person the one who is 'action calibrated for him.

The action is translated into a "command" for the user to go towards a specific node on the map, close to the person, in order to make him proceed towards a point of safety.

**- Movement execution**

Once the action to be performed has been obtained for each person, i.e. "the desired point to reach on the map", it is necessary to proceed to simulate the movement of this person.

The simulator is set up so that each single iteration of the process is equivalent to one second in the real world, and consequently the users for each iteration will proceed to move in the direction suggested by MARL proportionally to their speed, but always within one second.

This time interval made it possible to limit the "waiting times for a new action" that could arise in those cases in which users reached the destination nodes proposed by the action in less time than the simulated one.

Each person then moves towards their destination, rigorously following the real path that they would have available in the real world: therefore the virtual edge line that connects the two nodes is not considered, but the real path that represents the real roads and walkways of the city of Venice, and which also allows simulated users to bypass obstacles and natural elements on their path.

**- World status update**

At the end of the Movement Execution phase, i.e. when for each user a movement consistent with both the command received from the MARL and the surrounding environment has been simulated, the simulator then has at its disposal the updated arrangement of the people on the map after receiving the set of commands, and is able to proceed to a new iteration.

To do this, however, it is necessary to generate a two-dimensional array such that MARL is able to proceed with a new generation of commands. As described in the previous section, the MARL component uses an array containing for each node and each OSM edge the pair of data relating to:

- number of people present
- available bandwidth

Based on what was previously described, the simulator is already able to correctly calculate both of these values: the users, hitherto considered as separate atomic units, are then grouped based on their arrangement in the topography of Venice, obtaining the number of people present in each node and edge. Using the total number of people on each edge (which, remember, is an entity that can be assimilated to that of the street), updates the bandwidth available there.

Once this data set has been created, and therefore the state of the "world" within which the Simulator and MARL work within has been updated, the processing flow interfaces with the MARL again, sending the updated two-dimensional array.

With this data exchange, in the event that there are still users who need to be "saved", the MARL will proceed to generate new actions which it will then send to the Simulator, initializing a new iteration

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

Upon the activation of reinforcement learning for evacuation optimization, a collaborative framework integrating the UDT, simulation, and RL agent becomes operational. This section outlines the mechanisms of interaction and communication protocols among these key components, forming the core of the PER Use-Case MVP.

**Integration of Key Components**

The integration of the UDT, simulation, and RL agent is critical for the dynamic assessment and response to emergencies within the urban environment of Venice. The UDT acts as a central repository of real-time and historical urban data, including the physical layout of the city, current and projected crowd movements, and environmental conditions. The simulator, utilizing data from the UDT, creates detailed models of individual and crowd behaviors in response to various emergency scenarios.

This simulated environment provides a testing ground for the RL agent to evaluate different evacuation strategies without real-world consequences.

**Communication and Data Sharing Protocols**

Effective communication and data sharing between the UDT, simulator, and RL agent are facilitated through standardized protocols, primarily utilizing JSON for data interchange. This ensures that all components operate on consistent and up-to-date information. The RL agent receives updated city models from the UDT and simulated behavior patterns from the simulator, enabling it to refine its decision-making processes continually. Changes in the urban environment or evacuation strategies are promptly reflected across all platforms, allowing for a synchronized response to evolving situations. For inference purposes, the updated trained model is transparently propagated from training to inference via Sky Store Object Storage caching - see Deliverable D4.2.

This collaborative framework ensures that the PER Use-Case MVP can adapt to the complexities of managing evacuations in Venice's unique urban landscape. The seamless interaction between the UDT, simulation, and RL agent is foundational to developing effective evacuation strategies that are responsive to real-time changes in the urban environment and population dynamics.

### 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.

This systematic approach to training and operational deployment underscores the role of the RL agent in enhancing the responsiveness and adaptability of the PER Use-Case MVP.

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.

# 5. Next Steps

## TASKA Use Case MVP: Progress and Prospects

**Achievements:**

- *Data Processing Framework Established*: Implementation of a framework enabling definition and execution of scientific workflows.
- *Data Management Improvements*: Advanced handling capabilities for Measurement Sets (MS) data, facilitating efficient processing.
- *Automated Workflow Control*: Developed mechanisms allow seamless end-to-end data processing with minimal manual input.

**Ongoing Efforts:**

- *Workflow Refinement*: Continuous improvements to the workflow for enhanced robustness and user interaction capabilities.
- *Data Handling Enhancements*: Upgrading data set flexibility and robustness to accommodate varying volumes, ensuring efficient processing times.
- *User Interaction Features*: Improvement of user interface for more straightforward workflow management, facilitating easier task arrangement and data inspection.
- *Deployment on other infrastructures*: The TASKA-MVP is currently deployed on a dedicated testbed, waiting for the object-storage infrastructure to be available at ObsParis.

**Future Directions:**

- *Dataset Expansion*: Integration of larger, more varied data sets to test and refine processing capabilities.
- *Processing Task Development*: Addition of new processing tasks to extend the MVP's applicability within radio astronomy.
- *Community Collaboration*: Engagement with the scientific community to refine the MVP according to research requirements.
- *Broadened Scientific Integration*: Aiming to connect the TASKA MVP with wider scientific frameworks to increase accessibility and utility across different research domains.
- *Deployment on EOSC*: In addition to the current testbed and the ObsParis deployment, tests will be done to deploy the TASKA-MVP on EOSC.

The progression of the TASKA MVP is aimed at enhancing scientific data processing with an abstracted data management in a science application workflow. The other use-cases (see section 3.1) will also be studied:

- use-case A for real-time model serving of a classifier on Beam-form data (dynamic-spectra), which produces an event database (polygons in the temporal-spectra domain) and an optimized data storage strategy;
- use-cases C and D, for advanced imaging pipelines;
- use-case E, which gathers outputs from other use-cases into a merged event data base (temporal, spectral and spatial domainn).

*Interdisciplinary Applications*: Ongoing and future enhancements are intended to make the TASKA MVP adaptable to various scientific disciplines, facilitating a wider range of research applications. Collaboration with various stakeholders, including academic and research institutions, will be vital in exploring and implementing these applications.

## PER Use Case MVP: Progress and Prospects

The development of the MVP for the PER Use Case is advancing through essential stages.

Core achievements, ongoing projects, and future plans are delineated below.

**Achievements:**

*UDT Implementation:* The UDT integrates static and dynamic data, offering real-time situational awareness critical for emergency response optimization.
*MARL Advancements:* The MARL has been developed to improve the strategic planning of evacuation routes under variable urban scenarios.
*Simulation Environment Development:* A dedicated Simulator facilitates the safe testing and refinement of evacuation strategies, enhancing the predictive capacity and operational readiness of the MVP.

**Ongoing Efforts:**

*UDT Expansion:* Continuous enlargement of the UDT data pool to encompass broader urban dynamics, ensuring a more comprehensive emergency response.

*MARL Optimization*: Further refinement of MARL algorithms to increase accuracy and adaptability to complex evacuation scenarios.

*Simulator Enhancement:* Upgrading the simulation environment to address an expanded set of variables and emergency situations for improved realism and applicability.

**Future Directions:**

*EMA:* Plans are underway to incorporate the Evacuation Mobile App (EMA), aiming to provide real-time, user-specific evacuation guidance, significantly enhancing the interactive component of the MVP.

*Data Framework Expansion:* The Data Integration and Processing Framework will undergo enhancements to support greater data amalgamation and faster processing, pivotal for real-time decision-making.

*Community Engagement and Preparedness:* Execution of real life testing scenarios involving diverse participant groups to thoroughly assess system performance under varied conditions.

Through the deployment of the PER MVP, particularly the mobile app component, citizens become active participants in their safety. The system empowers individuals by providing them with timely information and personalized guidance during emergencies.
This approach fosters a greater sense of community preparedness and resilience, as residents and visitors are better informed and equipped to respond to potential threats.

*Cross-Sector Collaboration:* The ongoing development and refinement of the PER MVP are poised to offer substantial contributions to future urban planning and emergency management strategies. Insights gained from the Venice use case will inform broader applications of similar technologies in other urban settings, potentially transforming how cities worldwide approach emergency evacuations and urban safety.
Strengthening partnerships with municipal entities, technology vendors, and community organizations to foster a holistic emergency response ecosystem.

# 6. Conclusion

This document has detailed the initial deployment of the Minimum Viable Products for the PER and TASKA use-cases within the EXTRACT project framework. It delineates the progression from theoretical frameworks to operational implementations, directly addressing defined urban and scientific challenges.

The development of these MVPs has been grounded in extensive stakeholder engagement and a thorough analysis of use-case requirements. This ensures that the delivered solutions are not only technologically innovative but also pertinent and actionable within their intended contexts.

This phase represents a critical juncture in the EXTRACT project, demonstrating tangible advancements towards resolving complex challenges through technological innovation. The collaboration between various stakeholders and the adoption of a multidisciplinary approach underscore the project's comprehensive and methodical strategy.

As the first iteration of the MVPs, this deliverable is part of a larger, iterative development cycle. Insights gained and feedback received during this initial phase will drive future refinements and expansions. The project's commitment to continual improvement, guided by empirical data and stakeholder feedback, is fundamental to its methodology.

In summary, the release of the PER and TASKA MVPs constitutes an essential milestone in the EXTRACT project's trajectory. It establishes a concrete foundation for subsequent developments and reaffirms the project's ongoing commitment to delivering sophisticated, user-centric technological solutions in complex urban and research environments.

# 7. Acronyms and Abbreviations

- 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
- MARL - MultiAgent Reinforcement Learning Model
- MS - Measurement Sets
- MVP - Minimum Viable Product
- NRAO - National Radio Astronomy Observatory
- OSM - Open Street Map
- OTN - Open Transport Network
- PER – Personal Evacuation Route
- RDF - Resource Description Framework
- RFI - Radio interferences
- RL - Reinforcement Learning
- SCR - Smart Control Room
- TASKA - Transient Astrophysics using the SKA pathfinder
- UDT – Urban Digital Twin

# 8. References

[1] Annuario del turismo 2022 - Comune di Venezia: (https://www.comune.venezia.it/sites/comune.venezia.it/files/immagini/Turismo/Annuario_del_Turismo_dati_2022.pdf)