Digital environments – Version 1

Deliverable D1.2

Distribution level: Public

Version: 1.0

Date of delivery: 29/03/2025

AI4REALNET has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101119527, and from the Swiss State Secretariat for Education, Research and Innovation.

ai4realnet.eu

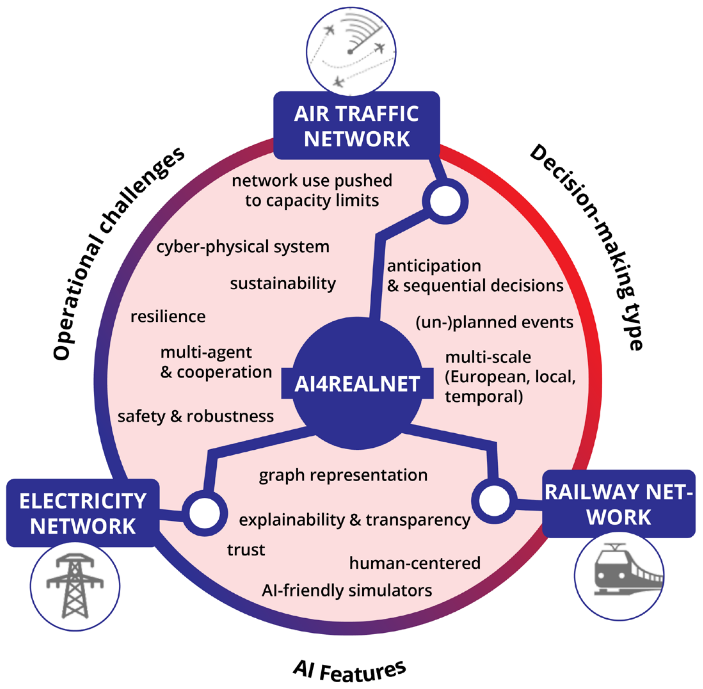
List of authors

Authors	Partner
Giulia Leto	TUD
Joost Ellerbrook	TUD
Clark Borst	TUD
Benjamin Donnot	RTE
Geoffroy Jamgotchian	RTE
Antoine Marot	RTE
Bruno Lemetayer	RTE
Maroua Meddeb	IRTSX
Milad Leyli-Abadi	IRTSX
Abderrahman Ait Said	IRTSX
Manuel Schneider	FLATLAND
Christian Eichenberger	FLATLAND
Manuel Meyer	FLATLAND

Background: AI4REALNET

GOALS

- Develop the next generation of decision-making methods powered by supervised and reinforcement learning, which aim at trustworthiness in AI-assisted human control, human-AI co-learning, and autonomous AI
- Boost the development and validation of novel AI algorithms via 3 existing open-source AI-friendly digital environments



List of software modules, data and resources

- Grid2op:
  - AI4REALNET/grid2op: platform to model sequential decision making in power systems
  - AI4REALNET/pypowsybl2grid: integration between Grid2op and PyPowSyBl backend (Python library for power grid modelling and simulation)
- Flatland:
  - AI4REALNET/flatland-rl: Simulation environment for modeling railway scheduling and rescheduling problems using reinforcement learning and operations research. Support for Gymnasium[1], RLlib[2], and PettingZoo[3] Environments. Generation of synthetic railway environments.
  - AI4REALNET/flatland-baselines: Baseline models for flatland.
  - AI4REALNET/flatland-scenarios: Scenarios for the railway use cases.
  - AI4REALNET/flatland-book: Conceptual and technical documentation of the Flatland environment.
- BlueSky:
  - AI4REALNET/BlueSky-Gym: A Gymnasium[1] style library for standardized Reinforcement Learning research in Air Traffic Management.
  - Use cases in BlueSky scenarios (proprietary data stored in the AI4REALNET GitLab, available upon request)
  - Scripted automation plugin in BlueSky: used for testing the integration with InteractiveAI, simulating sectorization (use case 1, use case 2) and re-routing around military areas (use case 2) for the Lisbon FIR
- InteractiveAI:
  - AI4REALNET/InteractiveAI : Generic platform for human interaction with complex networks
  - Specific HMI modules for each industrial domain (ATM, Power grid, Railway)
  - Interfaces with each digital environment (BlueSky, Grid2Op, Flatland)

[1] https://gymnasium.farama.org/
[2] https://docs.ray.io/en/latest/rllib/index.html
[3] https://pettingzoo.farama.org/index.html

Grid2Op – Power Grid

Main developer: RTE

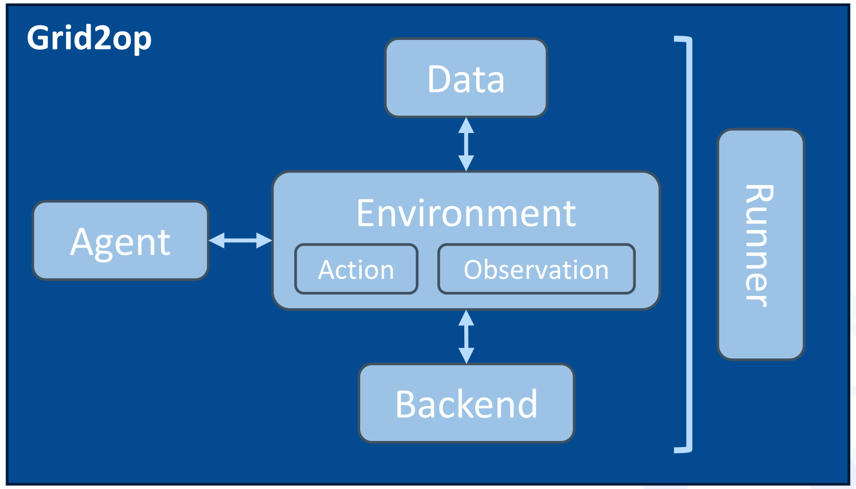
General description

Develop / train / evaluate performances of an agent acting on a Power grid

- Simulates model sequential decision making made by human operators to ease the research on sequential decision making applied to power systems
- Fully compatible with standard Gymnasium Reinforcement Learning API
- Abstracts the computation of the cascading failures
- Allows running same code with multiple power flows simulators (PandaPower, PyPowsybl, LightSim2Grid)
- Execute one agent on multiple independent scenarios in parallel
- Can be fully customized to serve different purposes (and not only Reinforcement Learning)
- Developed in Python
- Full documentation available at https://grid2op.readthedocs.io/en/latest/

Block diagram

- **Environment** is a representation of the power grid with which an Agent interacts (Generators, Loads, Powerlines, Shunts, Storage units, Substations)
- **Agents** can perform **Actions** on the environment, and receive **Observations** that correspond to its current or forecasted state
- Environment state evolves using **Data** (Time series) corresponding to injections (generators, loads) and structural information (planned outage, hazards)
- **Backend** allows calculating the physical state of the environment (Power grid, flow, intensity, etc.)
- The evaluation of agents can be performed by **Runner**, allowing corresponding data to be logged



Other experimental modules exists as part of the "GridAlive" ecosystem, but are not used in the project:

- Grid2Game (basic human GUI)
- Grid2Viz (visualization GUI)
- Grid2Bench (KPIs and benchmark)

Input data

- Environment
  - environment time series can be read from CSV (possibly by chunks to improve IO runtime)
  - “grid.json” is the full description of the powergrid characteristics, to be used by the backend
  - “grid_layout.json” is the geographic description of the powergrid for display
  - “config.py” is the configuration file
- Agent
  - Abstract class grid2op.Action.BaseAgent API can be implemented/extended to create custom agents
  - To perform their actions, Agents
    - receive grid2op.Reward.BaseReward and grid2op.Observation.BaseObservation data from the grid2op.Environment API
    - interacts with the grid2op.Action.ActionSpace API from the grid2op.Environment API
  - Abstract class grid2op.Action.BaseAction API can be implemented/extended to create custom actions
- Backend: new Backends can be implemented with grid2op.Backend.Backend API

Output data

- Environment
  - State of the environment is described by the grid2op.Observation.BaseObservation object
  - Observations are described by more than 30 attributes available in Observation API
  - Visualized using basic plotting features
    - Grid plot image can be generated using backend functionalities
    - Grid2Op allows for more advanced display features to be developed, for example in an advanced HMI framework
  - Exported in standard file formats (for example, iidm(*) thanks to the PyPowsybl backend)
- Runner : results of an evaluation are saved in standard JSON and NumPy (.npz) formats

(*) internal grid model initially developed under the iTesla research project that was funded by the European Union 7th Framework Programme (FP7)

Flatland - Railway

Main developer: Flatland Association

General description

The Flatland environment provides simplistic representation of a rail network on a grid world to address the vehicle rescheduling problem (VRSP)

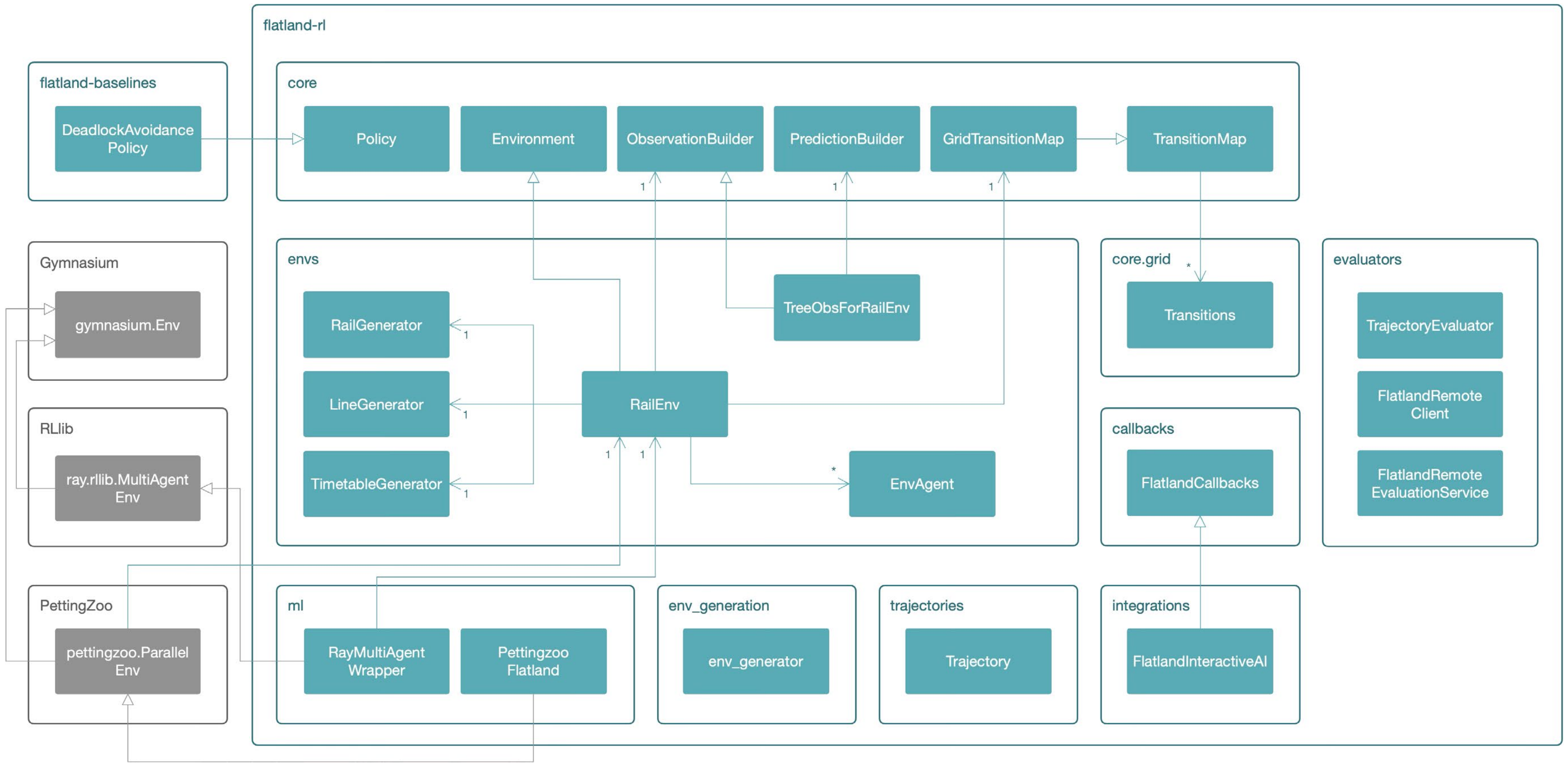
- Flatland is used to develop reinforcement learning (RL) solutions to the VRSP
- Trains are agents with a limited action space (↑ , ⟵ , ⟶ , ✕ , ⟳)
- Agents have schedules for their origin, destination and intermediate stops
- Railway network includes switches, slips, crossings and over-/underpasses
- Translation from grid representation of the network to a graph representation is implemented
- Agents can have speed profiles, reflecting different train classes (passenger, freight, etc.)
- Agents can be disrupted (in malfunction)

flatland-rl flatland-scenarios flatland-baselines flatland-book

General description

- Starter kit for easy access and as simple benchmark is provided
- Standard observations and rewards provided
- Trajectories are stored for benchmarking and regression testing
- Real-world example scenarios and generated railway environments are provided
- Integration with InteractiveAI API (user interface) provided
- Compatible with standard Gymnasium Reinforcement Learning API
- Compatible with RLlib and PettingZoo API
- Developed in Python
- Full documentation (Flatland book) available at: https://github.com/AI4REALNET/flatland-book

Block diagram



flatland-rl is the main package of Flatland and includes:

- **core** (base classes and functionality for simulations and reinforcement learning),
- **envs** (railway environment specific functionality),
- **callbacks** and **integrations** (for extended interactions, e.g., InteractiveAI),
- **ml** (Gymnasium, RLlib and PettingZoo support),
- **env_generation** and **trajectories** (environment generation and simulation run persistence)
- **evaluators** (functionality for scenario evaluation)

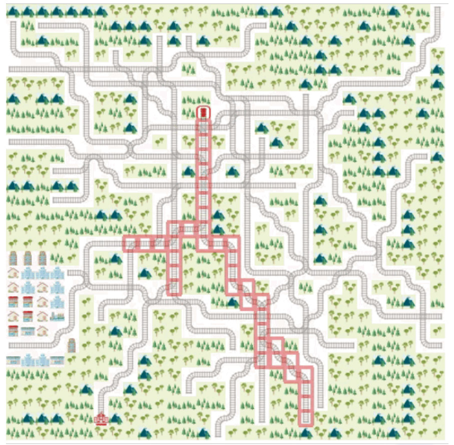
flatland-baselines provides a heuristic based policy baseline

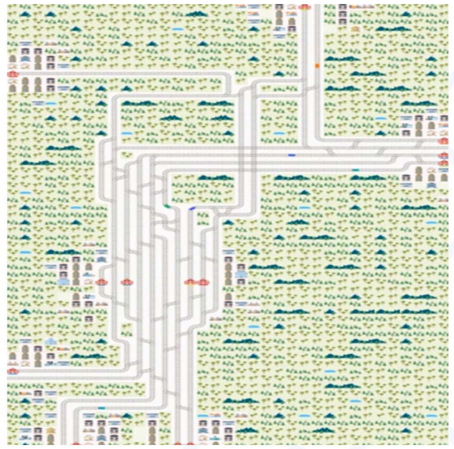
Input data

- Environment
  - Environments, i.e., railway network, train stations, schedules, lines, etc., can be specified manually or being generated based on parameters (grid size, number of agents, network characteristics, number of stations, etc.) in Python scripts.
  - Environments can be loaded from PKL (Pickled Python Objects) files.
  - Simulation runs, i.e., a sequence of environment states and actions (trajectories), can be loaded from PKL and TSV (tab-separated values) files.
- Agent
  - Agents interact with the environment by providing one of 5 actions (move forward, move left, move right, stop, do nothing) at each time step. With variable speed: acceleration by move forward action, deceleration by stop action.
  - Agents have access to an observation and reward at each time step.
  - Models for the Agents can be loaded from PKL files.
  - Gymnasium, RLlib, PettingZoo APIs are available for agent-environment interaction.

Output data

- Environment
  - Environments can be saved as PKL files.
  - State, observations and rewards are provided for each time step.
  - Trajectories can be saved as PKL and TSV files.
  - Renderings (videos / images) of simulation runs including observations and other information can be generated and saved in standard media file formats.
  - Events compatible with InteractiveAI framework can be exported or sent to live InteractiveAI instance.
- Agent
  - Actions can be exported as part of the trajectories.
  - Trained models can be saved as PKL files.
  - Training data is logged (e.g. rewards).


Visualization of the standard tree-observation


Real-world scenario demo (Olten, Switzerland): https://github.com/AI4REALNET/flatland-scenarios/raw/refs/heads/main/scenario_olten/img/olten_thumb.mp4

BlueSky – Air Traffic Management

Main developer: Delft University of Technology

BlueSky-Gym - General description

A Gymnasium style library for standardized Single-Agent Reinforcement Learning research in Air Traffic Management

- A platform to accelerate RL research in ATM, bridging gap between fundamental research and full-scale experiments
- Fully compatible with standard Gymnasium Reinforcement Learning API
- Compatible with RL libraries (StableBaselines-3, RLLib), allows for own algorithms
- Developed in Python

BlueSky-Gym - General description

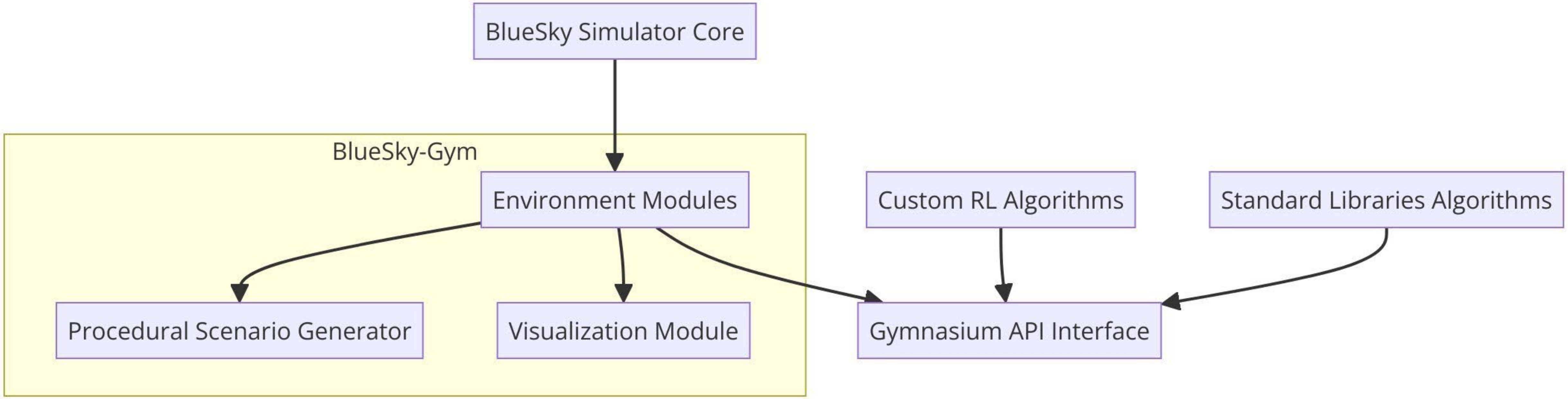
Environments:

- Environments included in the release can be used as benchmark for comparing results
- Scenarios are simplified, but representative of full-scale problems
- Customizable complexity and traffic types thanks to a modular design, providing standard observation and action spaces, reward functions for:
  - Horizontal environments for lateral navigation (observations: drift and distance from destination. Distance, bearing and speed with respect to intruders; actions: heading changes; reward: intrusions, drift, destination reached)
  - Vertical environments for vertical navigation (observations: altitude, vertical speed, target distance, runway distance. Bearing, speed and altitude with respect to intruders; actions: intrusion and target altitude distance, crash penalty)

Block diagram

- BlueSky Simulator Backend
  - Core air traffic simulation engine.
- Scripts to Procedurally Generate Scenarios
  - Generates diverse agents and initial conditions at each RL episode reset, following the logic of the specific scenario
- Gymnasium API
  - Defines states, actions, rewards, and episode control.
  - Compatible with standard RL libraries and custom RL algorithms
- Visualization module
  - PyGame based

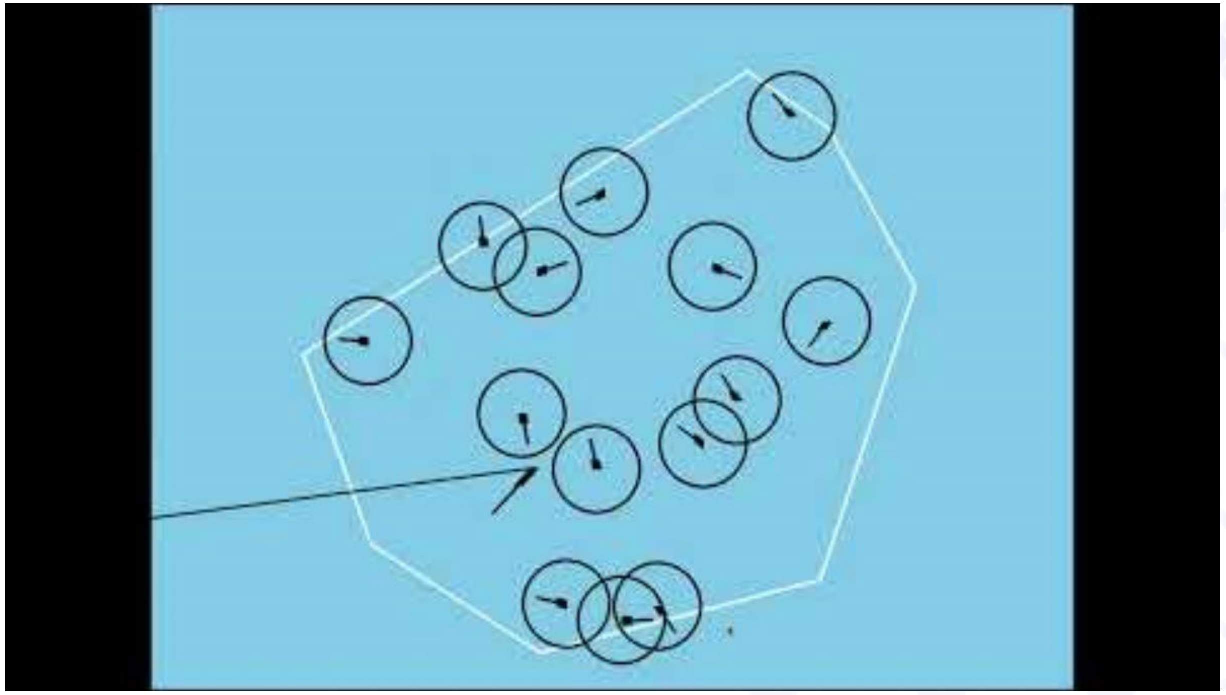
Block diagram



Use case 2 based environment

Avoiding restricted areas (military areas) in Lisbon FIR

- Static obstacles environment
  - Reach destination waypoint
  - Avoid obstacles



https://www.youtube.com/watch?v=EUxv4tQsMLs

Input data

- Environment
  - Encoded in Python scripts, generate a series of initial conditions for the BlueSky simulator.
- Agent
  - Based on BlueSky basic entities (traffic array)
  - To perform their actions, Agents interact with the environment through the Gym API, receiving observations and rewards

Output data

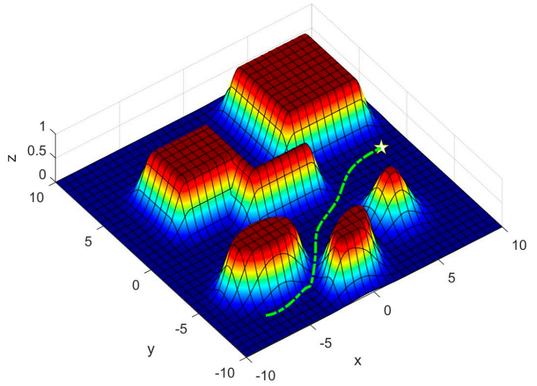
- Environment
    - State of the environment is advanced by the BlueSky backend
    - Observations
    - Conceptual visualizations customized each environment
    - Visualization of training progress
    - Observations and actions can be exported and read in BlueSky

- Trained model
    - parameters

Scripted automation - General description

1. Dynamic sector opening
   - Adapting sector opening plan based on sector count as a metric of workload
   - Sector count calculated from current occupancy in the altitude range of the sector + occupancy in the next 15 min based on required time of arrival

2. Avoidance of restricted regions
   - Randomly generated regions
   - Restricted regions detected in the flight plan by checking the upcoming two waypoints
   - Rerouting using potential fields and A*



(a) Potential field with a route, viewed from an angle.

Rivera et al., 2024

Input data

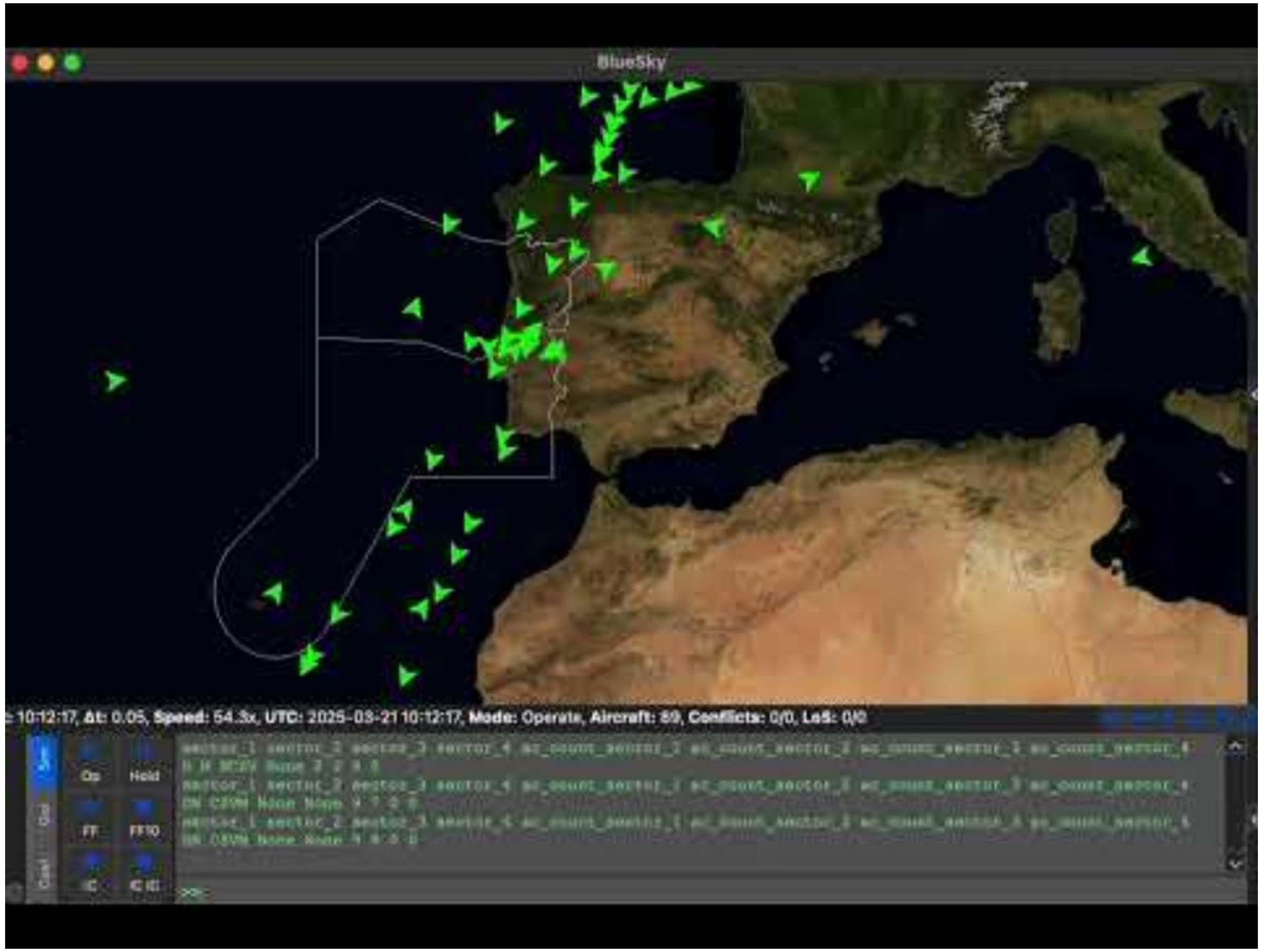
EUROCONTROL Aviation Data for Research, 2022-12-01

- Required Time of Arrival in BlueSky to mimic the flown trajectories
- All aircraft crossing Portuguese airspaces

Lisbon FIR sectors (UTA) from AIP

Easily adaptable to any other airspace

Output data



```
: 10:12:17, Δt: 0.05, Speed: 54.3x, UTC: 2025-03-21 10:12:17, Mode: Operate, Aircraft: 89, Conflicts: 0/0, LoS: 0/0
```

```
sector_1 sector_2 sector_3 sector_4 ac_count_sector_1 ac_count_sector_2 ac_count_sector_3 ac_count_sector_4
[illegible] None [illegible]
sector_1 sector_2 sector_3 sector_4 ac_count_sector_1 ac_count_sector_2 ac_count_sector_3 ac_count_sector_4
DN C3VN None None [illegible]
sector_1 sector_2 sector_3 sector_4 ac_count_sector_1 ac_count_sector_2 ac_count_sector_3 ac_count_sector_4
DN C3VN None None [illegible]
>>
```

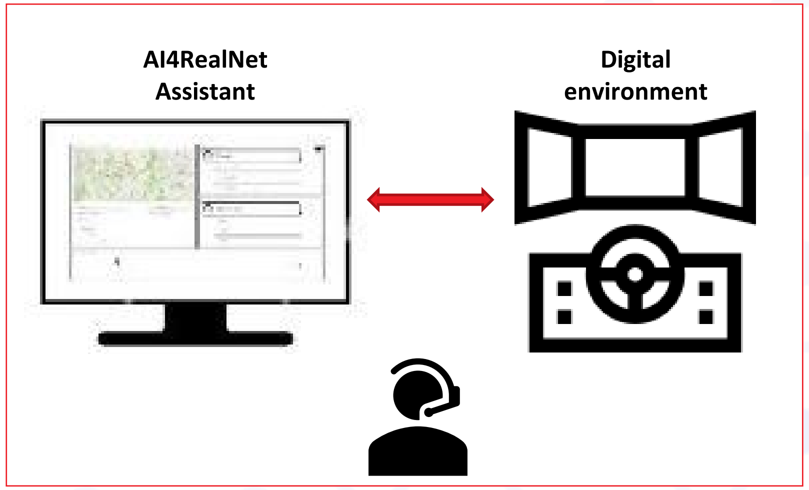
https://www.youtube.com/watch?v=ce00jTy--_o

InteractiveAI

Main developer: IRT SystemX

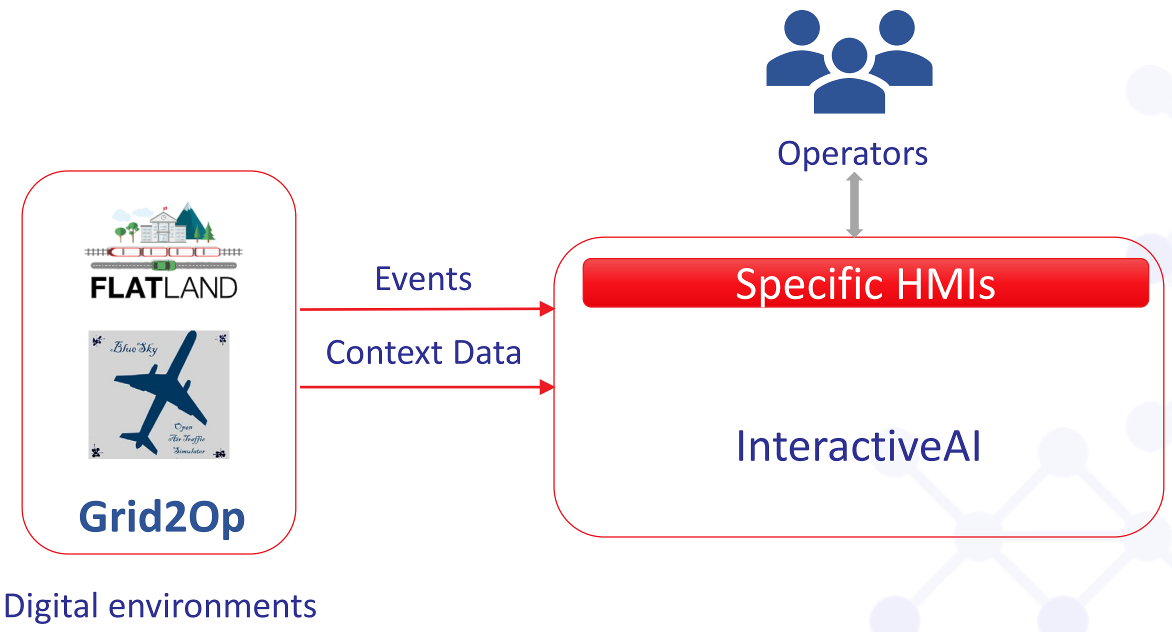
InteractiveAI Platform - Objectives

- The **InteractiveAI** platform is a **generic** technical base used to build interactive assistants for different industrial sectors
- The assistant objectives:
  - Display a user friendly context view about the digital environment : real time state
  - Notify the operator in case of incidents
  - Allow the operator to interact with his environment

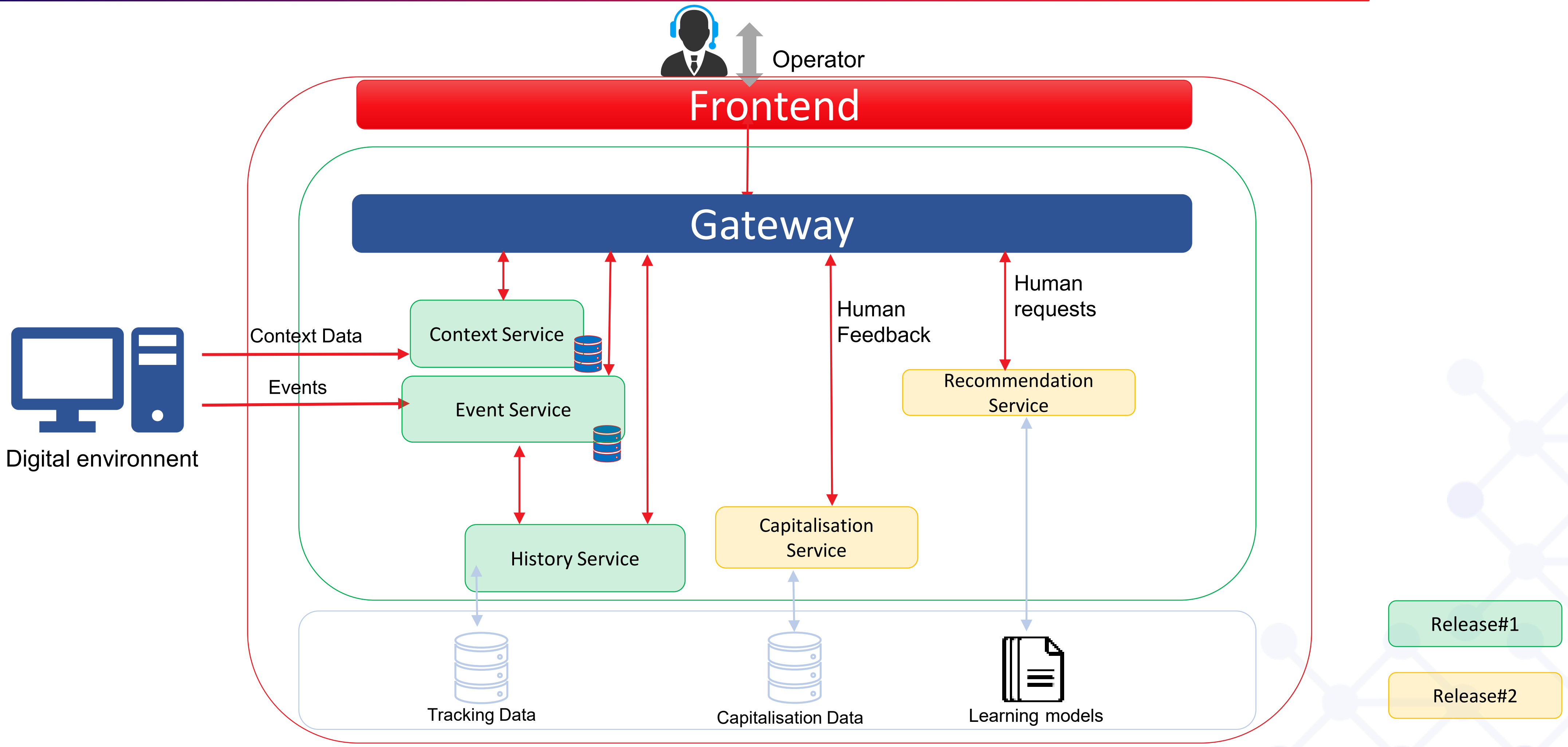


InteractiveAI Platform - Description

- **An interactive AI Assistant Platform for Real Time operations**
  - A generic technical base for all the industrial sector
  - Specific HMIs for each application domain
  - Communication with the digital environments



Block diagram

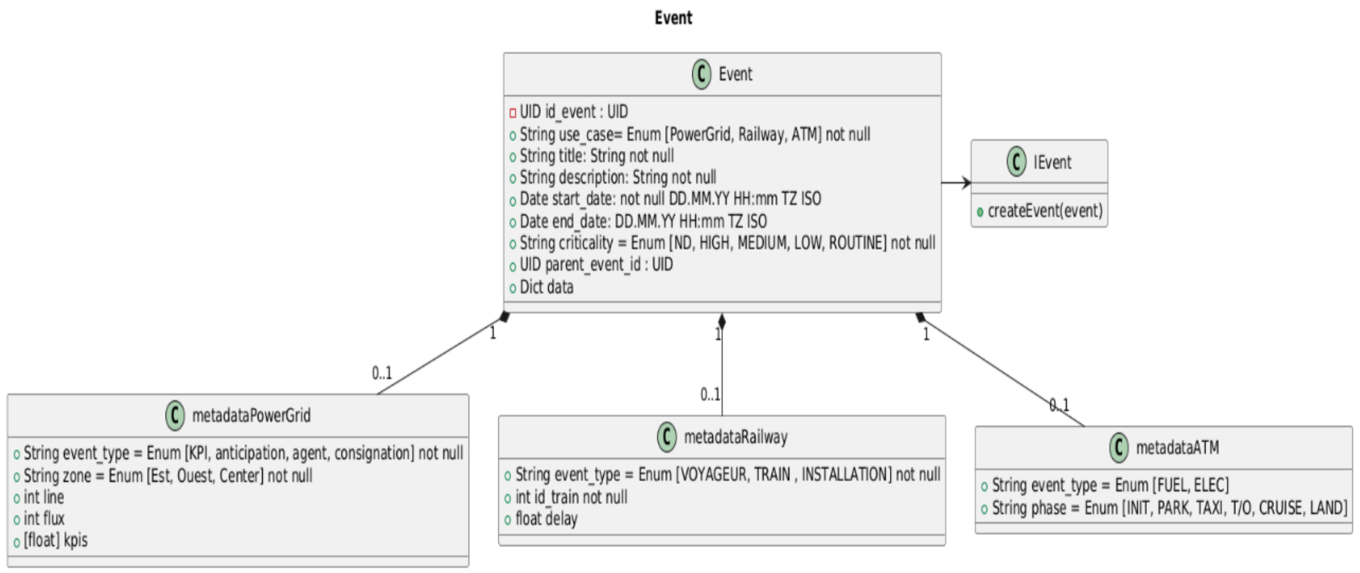


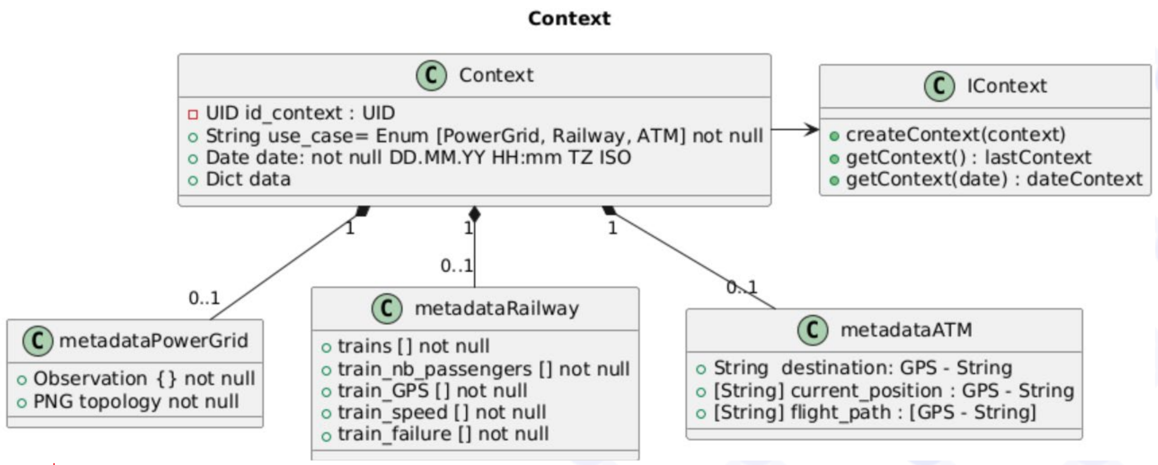
Inputs

- Different use case scenarios
- Specific needs for HMIs

Open source code

Inputs – Event and context APIs

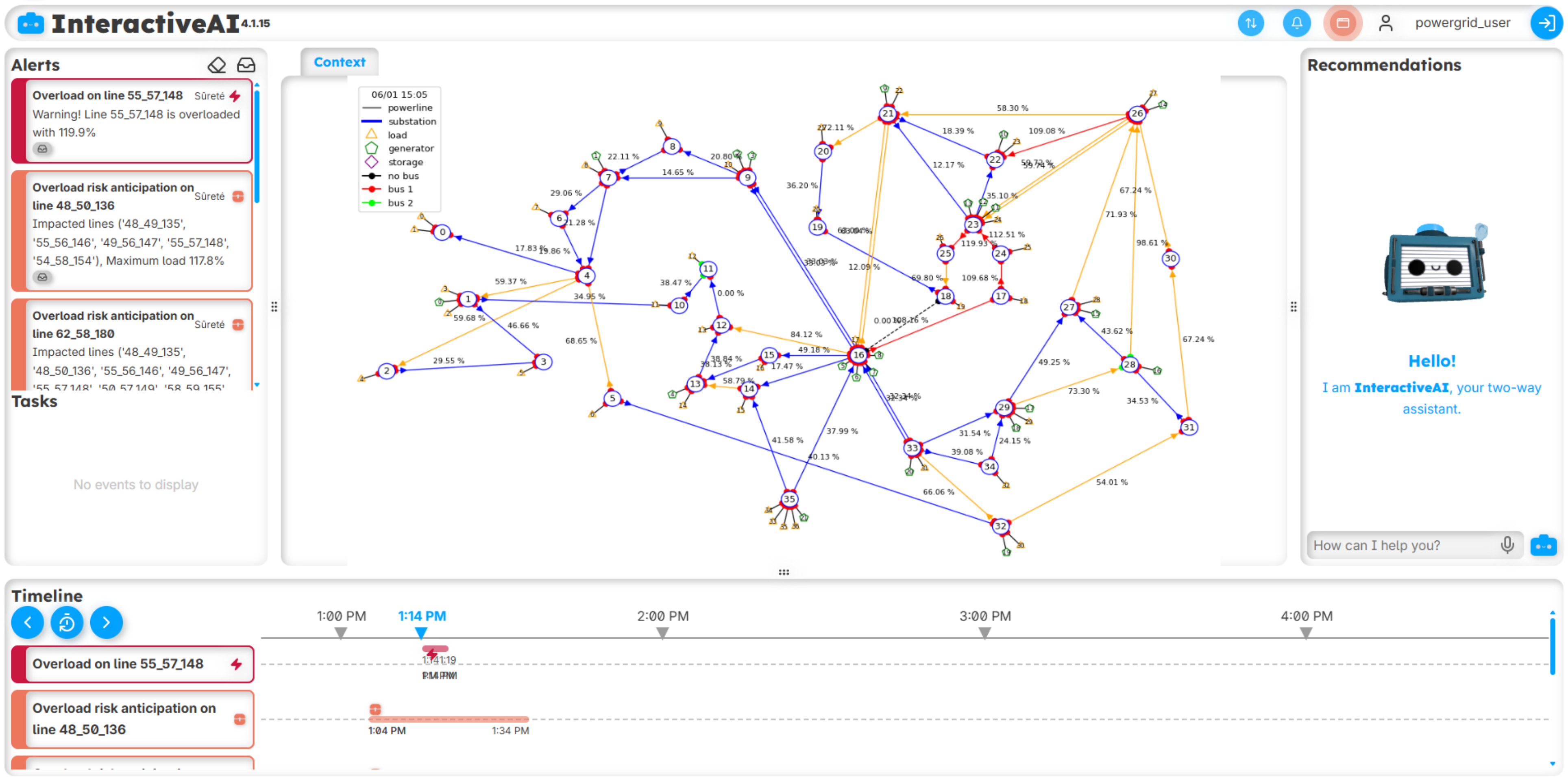

Event
C Event
UID id_event : UID
String use_case= Enum [PowerGrid, Railway, ATM] not null
String title: String not null
String description: String not null
Date start_date: not null DD.MM.YY HH:mm TZ ISO
Date end_date: DD.MM.YY HH:mm TZ ISO
String criticality = Enum [ND, HIGH, MEDIUM, LOW, ROUTINE] not null
UID parent_event_id : UID
Dict data
C IEvent
createEvent(event)
1
0..1
1
0..1
1
0..1
C metadataPowerGrid
String event_type = Enum [KPI, anticipation, agent, consignation] not null
String zone = Enum [Est, Ouest, Center] not null
int line
int flux
[float] kpis
C metadataRailway
String event_type = Enum [VOYAGEUR, TRAIN , INSTALLATION] not null
int id_train not null
float delay
C metadataATM
String event_type = Enum [FUEL, ELEC]
String phase = Enum [INIT, PARK, TAXI, T/O, CRUISE, LAND]


Context
C Context
UID id_context : UID
String use_case= Enum [PowerGrid, Railway, ATM] not null
Date date: not null DD.MM.YY HH:mm TZ ISO
Dict data
C IContext
createContext(context)
getContext() : lastContext
getContext(date) : dateContext
1
0..1
1
0..1
1
0..1
C metadataPowerGrid
Observation {} not null
PNG topology not null
C metadataRailway
trains [] not null
train_nb_passengers [] not null
train_GPS [] not null
train_speed [] not null
train_failure [] not null
C metadataATM
String destination: GPS - String
[String] current_position : GPS - String
[String] flight_path : [GPS - String]

Output - Graphical User Interface (GUI 1/3)

Power Grid

- Power Grid context:
  - Power grid network
  - Line flows
  - Overload - red lines
- Power Grid issues:
  - Overload
  - Risk anticipation
  - Different Criticality



Overload on line 55_57_148 Sûreté
Warning! Line 55_57_148 is overloaded with 119.9%

Overload risk anticipation on line 48_50_136 Sûreté
Impacted lines ('48_49_135', '55_56_146', '49_56_147', '55_57_148', '54_58_154'), Maximum load 117.8%

Overload risk anticipation on line 62_58_180 Sûreté
Impacted lines ('48_49_135', '48_50_136', '55_56_146', '49_56_147', '55_57_148', '50_57_149', '58_59_155'

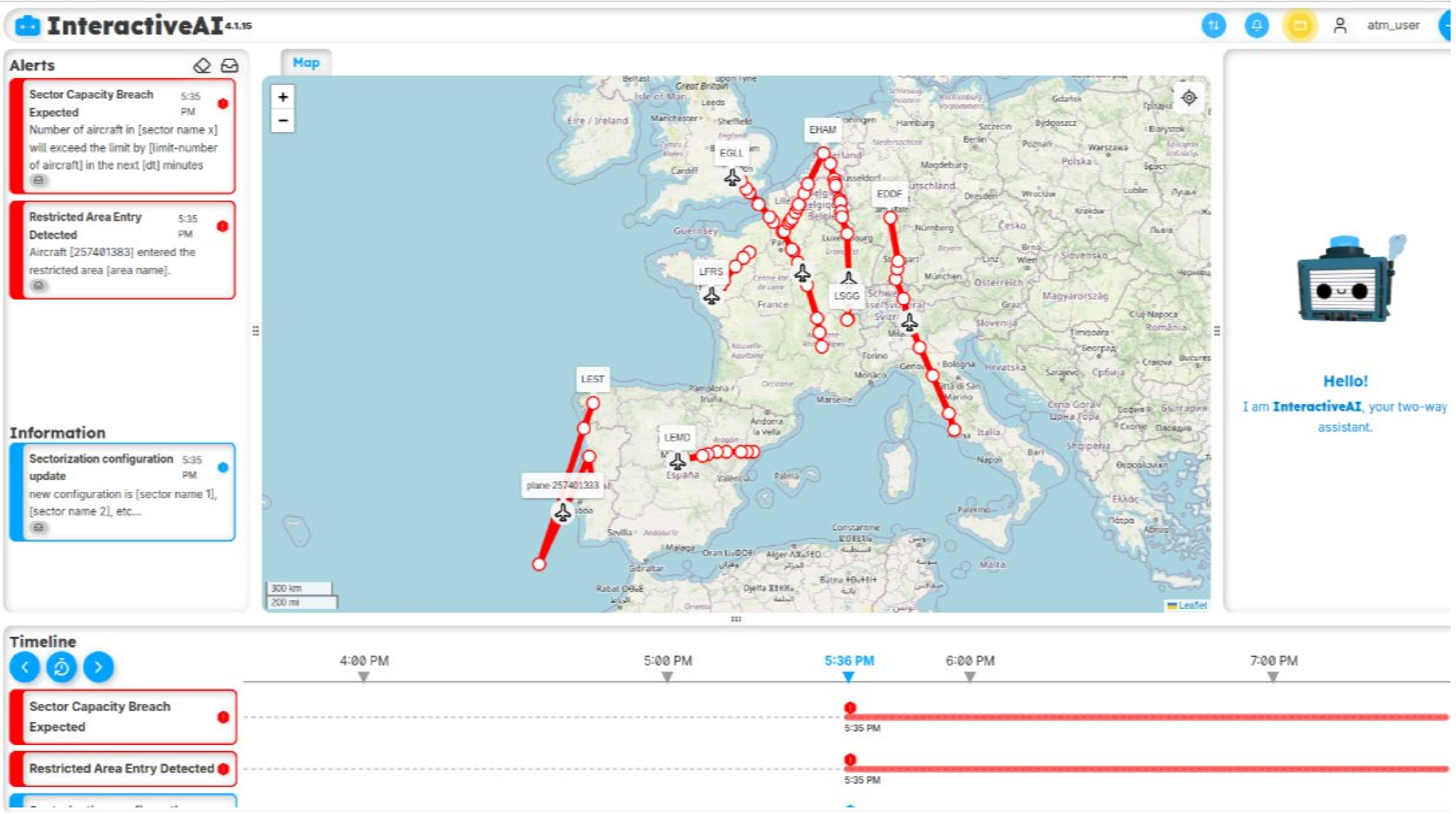
Overload on line 55_57_148

Overload risk anticipation on line 48_50_136
1:04 PM 1:34 PM

Output - Graphical User Interface (GUI 1/3)

ATM

- ATM context:
  - Planes' trajectories
  - Planes' GPS positions
- ATM issues:
  - Restricted Area entry detection
  - Sector capacity exeeded the maximum
- ATM informations:
  - Sectorization configuration update



Sector Capacity Breach Expected 5:35 PM
Number of aircraft in [sector name x] will exceed the limit by [limit-number of aircraft] in the next [dt] minutes

Restricted Area Entry Detected 5:35 PM
Aircraft [257401383] entered the restricted area [area name].

Sectorization configuration update 5:35 PM
new configuration is [sector name 1], [sector name 2], etc...

Hello!
I am **InteractiveAI**, your two-way assistant.

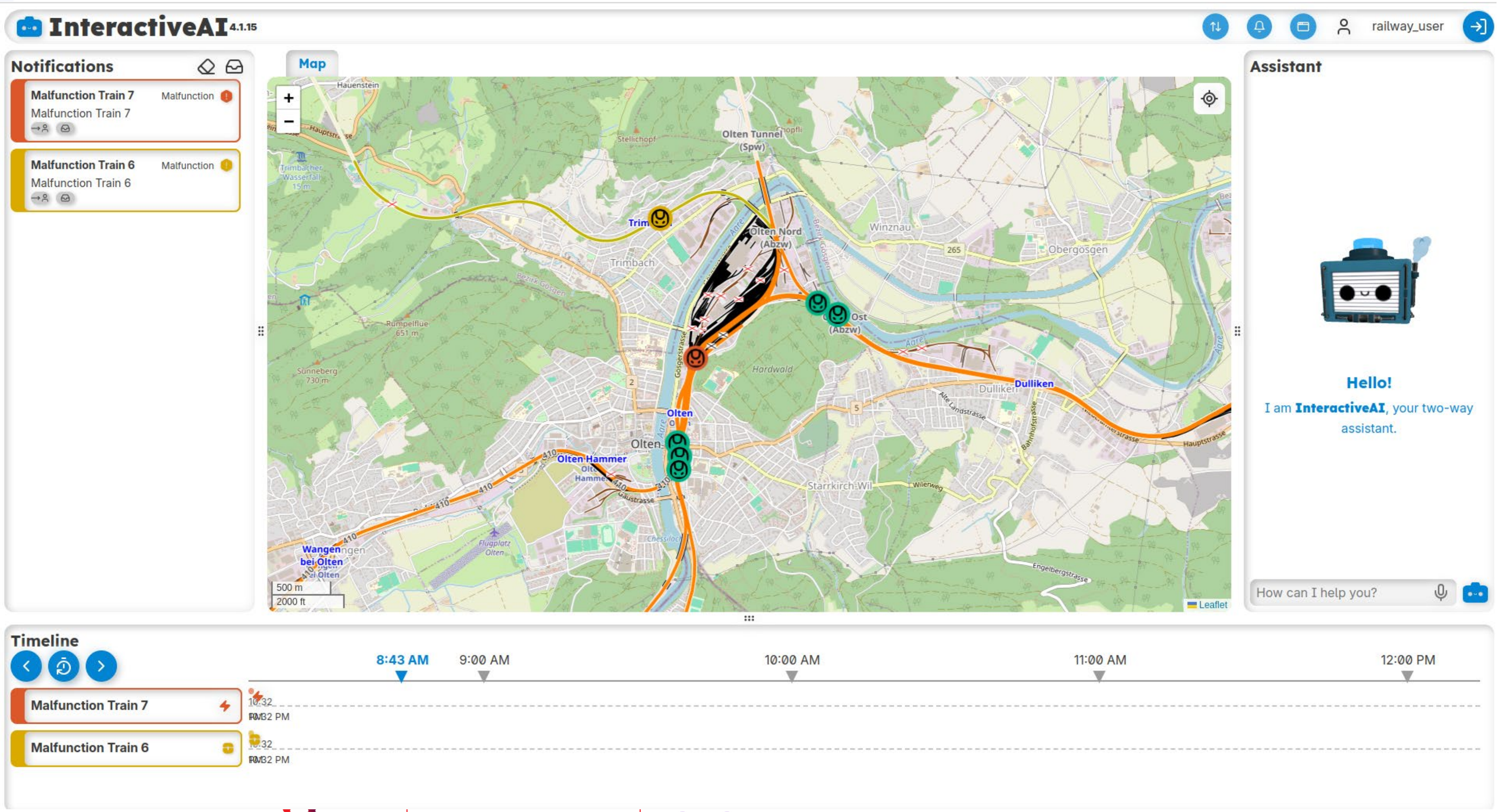
Sector Capacity Breach Expected

Restricted Area Entry Detected

Output - Graphical User Interface (GUI 3/3)

Railway

- Railway context:
  - Trains' GPS positions
  - Colored trains with issues
- Railway issues:
  - Malfunctions on trains
  - Different criticality


InteractiveAI 4.1.15 | railway_user

Notifications

Malfunction Train 7	Malfunction
Malfunction Train 7	
Malfunction Train 6	Malfunction
Malfunction Train 6	

Assistant

Hello!

I am **InteractiveAI**, your two-way assistant.

How can I help you?

Timeline

Malfunction Train 7

Malfunction Train 6

AI4REALNET has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101119527 and from the Swiss State Secretariat for Education, Research and Innovation