# Overview- Description- Purpose

A platform was designed and developed as part of Metropolis2 to support the concepts’ evaluation and trade off. The platform’s main scope is to read through all the data generated from the Metropolis2 simulations , reorganise them, process them to produce more complex data, compute the metrics values for all scenarios and finally create the graphs and diagrams to represent the calculated values.

# Setup and run

The platform is written in python and it is based on the utilisation of pandas dataframes to save the data and the computed metrics. The platform code may be found in <https://github.com/Metropolis-2/M2_data_analysis_platform> in the uop\_develop branch. The required dependencies to set up the environment can be found at the requirements.txt or the environment.yml of the current branch. The platform has been tested for python 3.9.12 .

To process new data you will have to run the code file named MainApp.py, which will create all the dataframes . While to create graphs from the dataframes, you will need to run the GraphCreator.py code.

The input files (log files and file intentions) for the platform are divided in four categories : Centralised, Decentralised, Hybrid and Fligth\_intentions and they should be placed in the regarding folders in the platform\_code/input\_logs folder.

The output dataframes are saved as dill files in the platform\_code/dills folder. The created graphs/diagrams are saved in the platform\_code/output\_graphs folder.

Before running the code you will have to check that all the necessary folders exist (if you cloned the repository/branch they should already be there but you will still need to add the input data= log files and flight intentions). The following checks will probably not be necessary, so you may skip them except if you get an error while trying to run the platform’s code. Here is a detailed description of what you will need to check:

1. If you want to process new logs to create the dataframes:
   1. You need a folder called “input\_logs”, which should contain four folders called: “Flight\_intentions”, “Centralised”, “Hybrid”, “Decentralised” and the log files and flight\_intention files should be placed in those folders. That folder is original in the “platform\_code” folder in which the MainApp.py code is also located. If your “input\_logs” folder is placed somewhere else you will have to change the path\_to\_logs variable at the beginning of the DataFrameCreator.py code to the path of your folder.
   2. You will need an empty folder called “dills” in which the created dataframes will be saved and it should be placed in the “platform\_code” folder in which the MainApp.py code is also located.
2. If you want to create graphs from already generated dataframes:
   1. You will need folder called “dills” in which the dataframes of interest should be placed. That folder is original in the “platform\_code” folder in which the GraphCreator.py code is also located. If your “dills” folder is placed somewhere else you will have to change the dills\_path variable at the beginning of the GraphCreator.py code to the path of your folder.
   2. To save the graph you will need a folder called “output\_graphs” in the “platform\_code” folder in which the GraphCreator.py. The output\_graphs folder has separate subfolders for each type of plot, so if you need to create a new output\_graphs folder copy the original.

**To run the overall system without mulitthreading:**

1. Place your input data in the corresponding folder
2. If you want to filter the log data for the first 1.5 hours simulation you may set the global variable time\_filtering to True , in the DataframeCreator.py code.
3. Run the MainApp.py
4. After the MainApp.py code is completed check that it created the dataframes. There should be 9 dill files in the dills folder.
5. Run the GraphCreator.py to create the graphs.

**To run the overall system with mulitthreading:**

1. Follow the procedure described above using the code found in the multithreaded\_platform branch.
2. In the code DataframeCreator.py , in the init function of the DataframeCreator class you will find the class variable threads. You may control the number of threads you want to use by setting that variable to the number you want.

# Basic architecture – IO

The platform inputs are the log files produced from the Metropolis2 simulations and the flight intentions files, describing the simulated scenarios. There are five types of log files: REGLOG, FLSTLOG, CONFLOG, LOSLOG and GEOLOG.

The platform loads the inputs files and saves the information of interest in pandas dataframes. There are two types of outputs produced; pandas dataframes saved as dill files and diagrams presenting the computed metrics.

The dataframes that the platform creates are the following:

* Flst\_log\_dataframe: is aggregated per flight and per scenario and contains data describing each flight from the FLSTLOG and the flight intention files.
* Loitering\_nfz\_datframe: is aggregated over the loitering nfzs and per scenario and contains data describing each loitering nfz applied from all the loitering aircraft during the simulations. The included data are generated based on data contained in the FLSTLOG and the flight intention files.
* Los\_log\_datframe: is aggregated per los event and per scenario and contains data describing each LOS event. The included data are extracted from the LOSLOG.
* Conf\_log\_datframe: is aggregated per conflict event and per scenario and contains data describing each conflict event. The included data are extracted from the CONFLOG.
* Geo\_log\_dataframe: is aggregated per geobreach and per scenario and contains data describing that geobreach event. The included data are generated by reading the GEOLOs and with the use of the Loitering\_nfz\_datframe and the Flst\_log\_dataframe.
* Env\_metrics\_dataframe: is aggregated per scenario and is includes the computed values for the ENV2 and ENV4 metrics for every scenario. The two metrics are calculated based on the data in the REGLOGs.
* Dens\_log\_datframe: is aggregated per time\_stamp and per scenario and includes the density (number of aircraft flying) in each time step. The density values are computed based on the REGLOG data.
* Metrics\_dataframe: is aggregated per scenario and it includes the computed values for all the metrics aggregated in a scenario level for every scenario. The metrics’ computations use all the datafarmes above.
* Prio\_metrics\_dataframes: is aggregated per priority level and per scenario and it includes values for the PRI3, PRI4, PRI5 metrics. The metrics’ computations use the flst\_log\_dataframe.

For the graphs’ generation the required dataframes are the Dens\_log\_datframe, Metrics\_dataframe and the Prio\_metrics\_dataframes .

# Metrics computation

|  |  |
| --- | --- |
| Metric | Computation methodology |
| Access and Equity |  |
| AEQ1 | Number of situations when realized arrival time of a given flight intention is greater than ideal expected arrival time by more or equal than some given cancellation delay limit that depends on mission type.  Computed as the number of aircraft that did not spawn plus the number of aircraft that spawned but did not arrive to their destination plus the number of aircraft that have an arrival delay larger than a threshold. The threshold in use is 5 minutes for emergency missions (priority 4), 15 minutes for not-loitering and not–emergency (priority<4) missions and 25 minutes for loitering missions. |
| AEQ1\_1 | AEQ1 / total number of aircraft described in the flight intention |
| AEQ2 | Number of situations when realized total mission duration is greater than specific drone autonomy.  Computed as the number of all the flights for which the flight\_time (as extracted from the FLST\_LOG is greater than the drone\_autonomy. The drone\_autonomy was set to 30 minutes. |
| AEQ2\_1 | AEQ2 / total number of spawned aircraft |
| AEQ3 | Measured as standard deviation of delay of all flight intentions, where delay for each flight intention is calculated as a difference between realized arrival time and ideal expected arrival time.  Computed by using the std() function of pandas over the arrival delay of all aircraft which arrived to their destination. |
| AEQ4 | Computed as the maximal difference between any individual flight intention delay and the average delay; where delay for each flight intention is calculated as the difference between realized arrival time and ideal expected arrival time.  The average delay was computed as the mean of the arrival delays of all aircraft which arrived to their destination. The metric is equal to the maximum **absolute** difference between the arrival delays of all aircraft which arrived to their destination and the computed average delay. |
| AEQ5 | Number of flight intentions whose delay is greater than a given threshold from the average delay in absolute sense.  The average delay was computed as the mean of the arrival delays of all aircraft which arrived to their destination.  The metric is computed as the number of aircraft that did not spawn plus the number of aircraft that did not arrive to their destination plus the number of aircraft that arrived in their destination in time and the absolute difference between their arrival delays and the average delay is larger than a threshold. The threshold is set to 50 seconds. |
| AEQ5\_1 | AEQ5 / total number of aircraft described in the flight intention |
| Capacity |  |
| CAP1 | Measured as an arithmetic mean of delay of all flight intentions, where delay for each flight intention is calculated as the difference between realized arrival time and ideal expected arrival time.  Computed as the mean of the arrival delays of all aircraft which arrived to their destination. |
| CAP2 | Number of intrusions per flight intention I.e., a ration between total number of intrusions (SAF-2 indicator) and number of flight intentions.  SAF2/ total number of spawned aircraft |
| CAP3 | Computed only for scenarios with the rogue uncertainty.  CAP3=CAP1\_for rogue scenario –CAP1\_for regarding baseline scenario |
| CAP4 | Computed only for scenarios with the rogue uncertainty.  CAP4=CAP2\_for rogue scenario –CAP2\_for regarding baseline scenario |
| Efficiency |  |
| EFF1 | Ratio representing the length of the ideal horizontal route to the actual horizontal route.  Only aircraft that arrived in their destination were considered. The sum of the baseline\_2d\_distance divided by the sum of the actual\_2d\_distance. |
| EFF2 | |  | | --- | | Ratio representing the length of the ideal vertical route to the actual vertical route. |   Only aircraft that arrived in their destination and had a valid value for actual\_vertical\_distance were considered. The sum of the baseline\_vertical\_distance divided by the sum of the actual\_vertical\_distance. |
| EFF3 | Ratio representing the length of the ascending distance in the ideal route to the length of the ascending distance of the actual route.  Only aircraft that arrived in their destination and had a valid value for actual\_ascending\_distance were considered. The sum of the baseline\_ascending\_distance divided by the sum of the actual\_ascending\_distance. |
| EFF4 | Ratio representing the 3D length of the ideal route to the 3D length of the actual route.  Only aircraft that arrived in their and had a valid value for actual\_3d\_distance were considered. The sum of the baseline\_3d\_distance divided by the sum of the actual\_3d\_distance. |
| EFF5 | Ratio representing the time duration of the ideal route to the time duration of the actual route.  Only aircraft that arrived in their destination were considered. The sum of the baseline\_flight\_time divided by the sum of the actual\_fligth\_time. |
| EFF6 | Time duration from the planned departure time until the actual departure time of the aircraft.  Computed as the mean of the departure delay for all aircraft that arrived in their destination. |
| Safety |  |
| SAF1 | Number of aircraft pairs that will experience a loss of separation within the look-ahead time.  The number of conflicts that occurred. |
| SAF1\_2 | Number of conflicts per flight.  SAF1/number of spawned aircraft |
| SAF2 | Number of aircraft pairs that experience loss of separation  The number of los events that occurred. |
| SAF2\_1 | Number of severe intrusions.  The number of los events which would cause a crash. |
| SAF3 | Ratio representing the proportion of conflicts that did not result in a loss of separation.  (saf1-saf2)/saf1 |
| SAF4 | The minimum separation between aircraft during conflicts.  The minimum value of the aircraft distance during the LOS events that occurred. |
| SAF5 | Total time spent in a state of intrusion.  The sum of the time spend in LOS over all LOS events that occurred. |
| SAF6 | The number of geofence/building area violations.  The number of geobreaches that occurred. |
| SAF6\_1 | Number of severe geofence violations.  The number of severe geobreaches that occurred. A severe geobreach is defined as a geobreach that violated for over 1 meter. |
| SAF6\_2 | Number of severe loitering NFZ violations.  The number of severe geobreaches in loitering nfzs that occurred. |
| SAF6\_3 | Number of severe buildings/static geofences violations.  The number of severe geobreaches in not loitering nfzs that occurred. |
| SAF6\_4 | Number of severe open airspace geofences violations .  The number of severe geobreaches in open airspace that occurred. |
| SAF6\_5 | Number of severe buildings violations .  The number of severe geobreaches in building geofences that occured. |
| SAF6\_6 | Number of severe loitering NFZ violations, with origin/destination in NFZ.  The number of severe geobreaches in loitering nfzs , in which the violating aircraft had its origin or destination node in the nfz, that occurred. |
| SAF6\_7 | Number of severe loitering NFZ violations within 3 minutes of the NFZ activation.  The number of severe geobreaches in loitering nfzs , in which the violating aircraft was already in the nfz area when the nfz was applied, that occurred. The aircraft is considered to already be in the nfz area if the logged geobreach time is less than 3 minutes after the application time of the nfz. |
| Environmental |  |
| ENV1 | Work done, computed as the sum of work done of every aircraft that spawned and had a valid value for vertical distance (ALT\_DIST in flst\_log). The work done per aircraft is computed as flight time +ascending distcane/vertical speed |
| ENV2 | Average flight level weighed by the length flown at each flight level.  The sum of all the length of the route segments of all the aircraft as logged in the reglog multiplied by the altitude (in metetrs) of the segment, divided by the sum of the of all the length of the route segments of all the aircraft as logged in the reglog. |
| ENV3\_1 | Sound exposure.  Represent total sound exposure at the given pointσ on city area surface. It is computed by aggregating the total sound intensity (of all sound sources) at that given point over the time. |
| ENV3\_2 | Number of points with significant sound exposure.  Number of situations where the total sound intensity at a given point at a given time stamp is more than the sound exposure threshold. Sound exposure threshold to be decided. |
| ENV4 | The ratio between the difference of maximum and minimum length flown at a flight level and average length flown at level.  Flight levels in use: starting at 30 feet and going up to 480 with step 30. |
| Priority |  |
| PRI1 | Total duration of missions weighted in function of priority level.  Only aircraft that arrived in their destination were considered. The sum of flight time of the aircraft multiplied by a weight dependant on their priority. The weight is 1 for priority 1 , 2 for priority 2, 4 for priority 3 and 8 for priority 4. |
| PRI2 | Total distance travelled weighted in function of priority level.  Only aircraft that arrived in their destination were considered. The sum of the 3d distance of the aircraft multiplied by a weight dependant on their priority. The weight is 1 for priority 1 , 2 for priority 2, 4 for priority 3 and 8 for priority 4. |
| PRI3 | The average mission duration for each priority level per aircraft.  Computed separately for every priority level. Only aircraft that arrived in their destination were considered. The mean of the flight time of the aircraft of the specific priority level. |
| PRI4 | The average distance travelled for each priority level per aircraft.  Computed separately for every priority level. Only aircraft that arrived in their destination were considered. The mean of the 3d distance of the aircraft of the specific priority level. |
| PRI5 | The total delay experienced by aircraft in a certain priority category relative to ideal conditions.  Computed separately for every priority level. Only aircraft that arrived in their destination were considered. The sum of the arrival delay of the aircraft of the specific priority level. |