Analysis & report, integration

Deliverable ID:	D5.1
Dissemination Level:	PU
Project Acronym:	MAHALO
Grant:	892970
Call:	H2020-SESAR-2019-2
Topic:	SESAR-ER4-01-2019
Consortium Coordinator:	DBL
Edition date:	30 November 2021
Edition:	00.01.00
Template Edition:	02.00.02

Authoring & Approval

Authors of the document

Name/Beneficiary	Position/Title	Date
Erik-Jan van Kampen/TUD	WP5 Leader	10/11/2021
Tiago Monteiro Nunes/TUD	WP5 Researcher	15/11/2021
Magnus Bång/LIU	WP5 Researcher	17/11/2021
Sami Yahia/LIU	WP5 Researcher	22/11/2021

Reviewers internal to the project

Name/Beneficiary	Position/Title	Date
Carl Westin (LiU)	WP6 Leader	22/11/2021
Brian Hilburn (CHPR)	WP2 Leader	22/11/2021
Clark Borst (TUD)	WP4 Leader	23/11/2021
Martin Christiansson (LFV)	Project member	24/11/2021
Matteo Cocchioni (DBL)	Project Contributor	26/11/2021
Stefano Bonelli (DBL)	Project Coordinator	26/11/2021

Approved for submission to the SJU By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date
Stefano Bonelli (DBL)	Project Coordinator	26/11/2021

Rejected By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date

Document History

Edition	Date	Status	Author	Justification
00.00.01	10/11/2021	Draft	Erik-Jan van Kampen	Doc created
00.00.02	12/11/2021	Draft and internal review	Erik-Jan van Kampen	Deliverable improvement
00.00.03	15/11/2021	Draft and internal review	Tiago Monteiro Nunes	Deliverable improvement
00.00.04	17/11/2021	Draft and internal review	Magnus Bång	Deliverable improvement
00.00.05	22/11/2021	Draft and internal review	Sami Yahia	Deliverable improvement
00.00.06	26/11/2021	Internal release	Matteo Cocchioni	Deliverable to MAHALO management
00.01.00	26/11/2021	Final Release	Stefano Bonelli	Deliverable approved for submission

Copyright Statement

MAHALO

MODERN ATM VIA HUMAN / AUTOMATION LEARNING OPTIMISATION

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 892970 under European Union's Horizon 2020 research and innovation programme.

Abstract

This document describes the efforts performed on the integration of Machine Learning models from MAHALO WP3 with the Human Machine Interfaces from MAHALO WP4, as part of MAHALO WP5. A framework is presented that is used to test the integration of the components, which contains a number of interfaces where information is transferred between ATC simulators, displays and machine learning models. First the overall pipeline of this integration testing framework is presented. After that the interface and data formats are presented.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Placement of WP5: Integration within MAHALO

MAHALO asks a simple but profound question: in the emerging age of Machine Learning (ML), should we be developing automation that matches human behaviour (i.e., conformal), or automation that is understandable to the human (i.e., transparent)? Further, what trade-offs exist, in terms of controller trust, acceptance, and performance?

To answer these questions, two types of machine learning models have been developed in WP3 [1]: a conformal/personalized prediction model based on Supervised Learning (SL) and an optimized prediction model based on Reinforcement Learning (RL).

Figure 1 gives an overview of the MAHALO work packages and how they are connected. In WP4 [2] an Ecological User Interface (E-UI) has been developed that can present the operator with ATC conflicts and that can process input from the operator that is aimed to resolve the conflict. The E-UI has two main purposes. Firstly the E-UI is used for data collection, where ATCO resolutions are logged for specific conflict scenarios. The second purpose of the E-UI is to present the ATCO with the advisories coming from the Machine Learning models and to get feedback from the ATCO about the level of agreement with the proposed resolution. The E-UI also has to be able to present different levels of transparency when providing the advisory from the Machine Learning model.

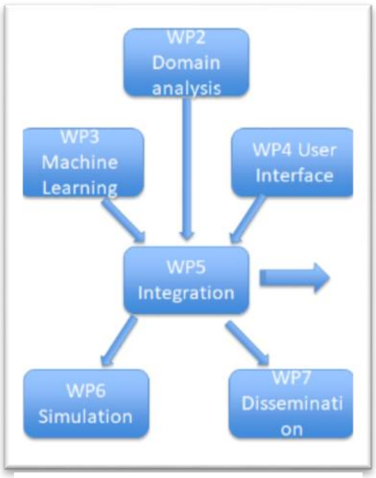


Figure 1: MAHALO Work Packages

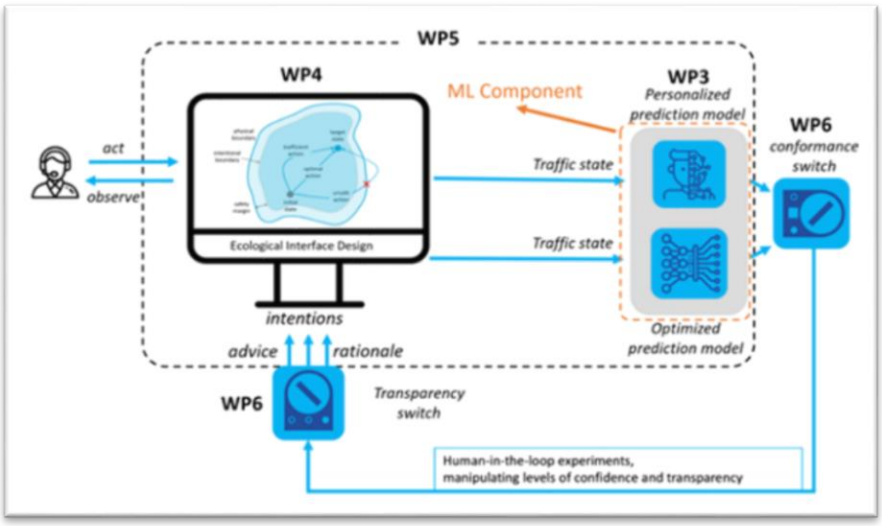

WP5
WP4
ML Component
WP3
Personalized prediction model
WP6
conformance switch
act
observe
Traffic state
Traffic state
Ecological Interface Design
intentions
advice
rationale
Optimized prediction model
WP6
Transparency switch
Human-in-the-loop experiments, manipulating levels of confidence and transparency

Figure 2: Placement of WP5

Figure 2 shows how the Machine Learning models from WP3 and the E-UI from WP4 interact within the whole MAHALO framework. From this figure, the placement of WP5 also becomes clear. WP5 will integrate the ML models and the E-UI, which were initially developed independently.

Since WP5 is about integration, it mostly contains technical tasks, such as defining interfaces and data formats. It is a crucial prerequisite for having the main MAHALO experiments with ATCOs in WP 6.

1.2 Objective

The objective of WP5 is to test the integrated capabilities of the ML+E-UI combination, such that the complete system will be ready for the experiments with real ATCO's, which will take place in WP6. The objective of this report is to describe the Integration process and outcomes. It also serves as a reference for data formats and interface designs between the different components in the system.

1.3 Scope

WP5 covers the integration of the ML models and E-UI, which will be demonstrated in an experiment, SIM1. It should be noted that SIM 1 is **not** meant to draw conclusions about the effect of ML on ATCO performance or acceptance, which will be evaluated in the main experiments SIM2a and SIM2b in WP6. Because the goal of SIM 1 is only to test integration and interfaces, it is not necessary for SIM 1 to be conducted by real air traffic controllers. The outcomes of SIM 1 will be able to show that the integrated system is ready to be used by real air traffic controllers. The ML models generated in SIM 1 are not expected to be of high quality because of the limited number of samples used for training and the quality of those samples, i.e. data generated by non-experts is possibly inconsistent, which is detrimental for ML training. SIM1 itself is documented in MAHALO deliverable D5.2.

1.4 Report structure

Chapter 2 of this document contains the main overview of the WP5 integration procedures. It describes the integration pipeline and identifies where interfaces need to be defined, which will be discussed in Chapter 3. Besides the interface definitions, Chapter 3 also contains definitions of the main information types/data that is stored and transferred in the integrated system.

Chapter 4 presents the main conclusions and outcomes of the integration process in WP5, with a summary of achieved results and lessons learned.

2 Integration procedure: Machine Learning models and Ecological User Interface

In order to integrate the ML models and the E-UI, several interfaces have been identified where software from different parts of the system needs to communicate with other parts of the software or hardware. Figure 3 and Figure 4 show flowcharts of intended operation of the Optimized and Conformal ML models. The flowcharts are split into several steps, indicated by the blue dotted borders. The steps are described in more detail in The other components in Figure 3 and Figure 4 are data inputs/outputs and converters that convert data files from one format to another.

Table 1.

The main items in these flowcharts are:

• Bluesky[1]	An ATC simulator that is used to run faster-than-real-time simulations
• SectorX	The ATC simulator that is used for the human-machine interaction.
• RL Model	The optimized ML model based on Reinforcement Learning
• CNN/SL Model	The conformal ML model based on Supervised Learning

[1] http://homepage.tudelft.nl/7p97s/bluesky/

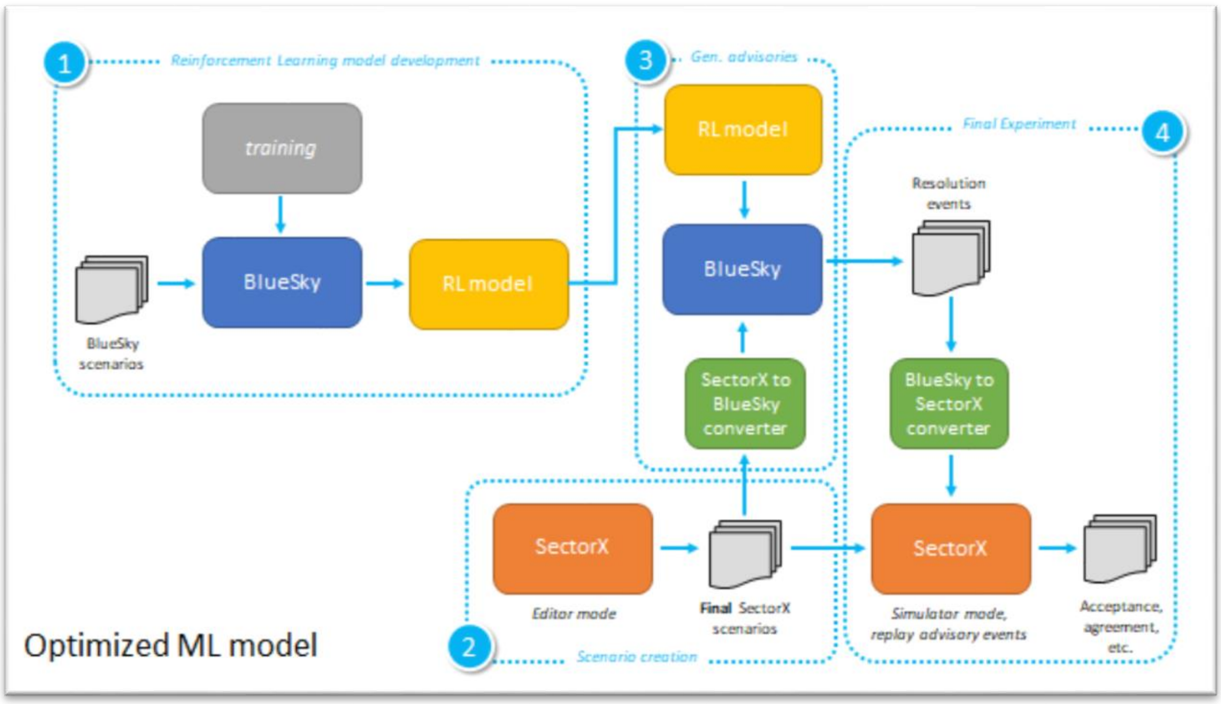


Figure 3: Optimized ML model flowchart SIM 1

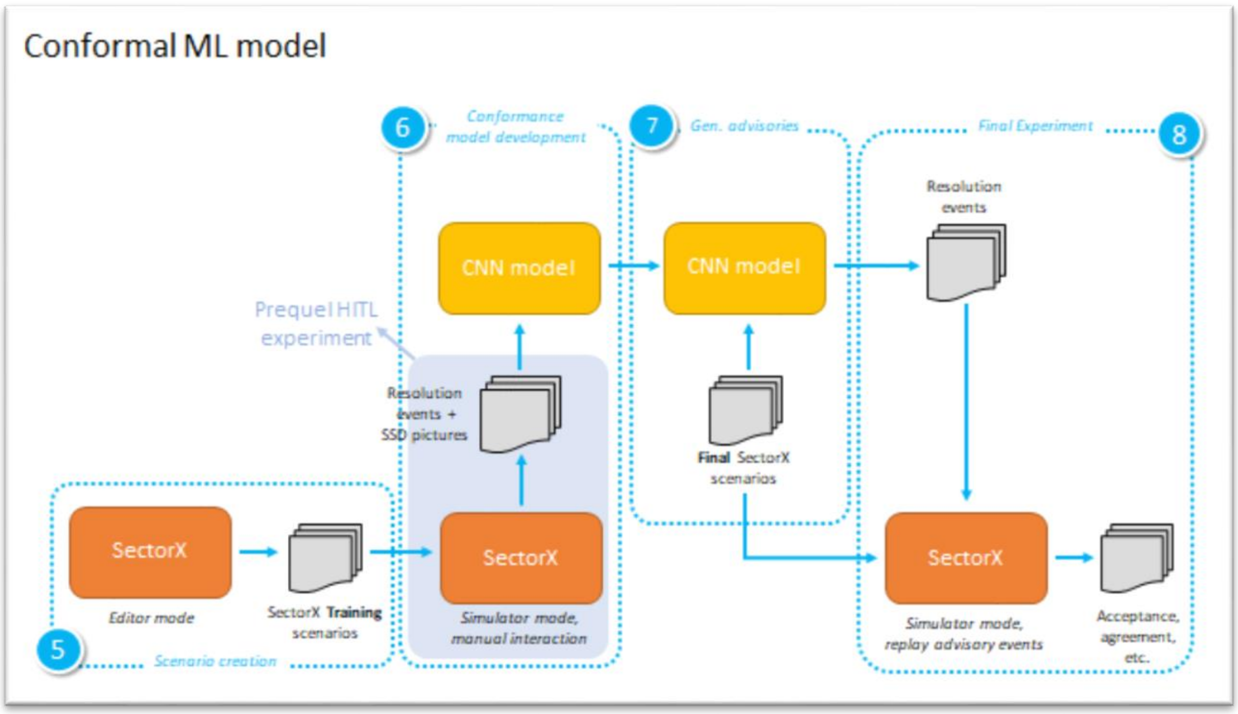

Conformal ML model
6
Conformance model development
7
Gen. advisories
Final Experiment
8
CNN model
CNN model
Resolution events
Prequel HITL experiment
Resolution events + SSD pictures
Final SectorX scenarios
SectorX
Editor mode
SectorX Training scenarios
SectorX
Simulator mode, manual interaction
SectorX
Simulator mode, replay advisory events
Acceptance, agreement, etc.
5
Scenario creation

Figure 4: Conformal ML model flowchart SIM 1

The other components in Figure 3 and Figure 4 are data inputs/outputs and converters that convert data files from one format to another.

Table 1: Pipeline description for the integration test frameworks from Figure 3 and Figure 4

Step	Description
1	The Bluesky ATC simulator is used to train the RL agent. The RL agent interacts with the Bluesky simulator in many (>40k) automatically generated scenarios with different geometries and learns what is the best strategy for a given scenario description.
2	The SectorX ATC simulator is used to generate a small number of hand-crafted scenarios that will be presented to the human air traffic controller together with the resolution provided with the RL agent.
3	The hand-crafted SectorX scenarios are converted to Bluesky scenarios using an automated script (see Section 3.1.4). These scenarios are then presented to the trained RL agent. This trained RL agent then provides resolutions/advisories, which are stored.
4	The stored resolutions/advisories are converted into SectorX event-logs. These event-logs can used, together with the original SectorX scenarios from step 2, to playback the scenario in SectorX and to present the resolutions/advisories that were generated by the RL agent to the human air traffic controller. Several options for providing transparency of the generated resolutions/advisories will be provided. The human air traffic controller can evaluate the proposed resolutions/advisories and accept or reject them. Also feedback can be given on the reason of acceptance/rejection.
5	The SectorX ATC simulator is used to generate training scenarios that will be presented to the human air traffic controller.
6	The SectorX ATC simulator is used to generate training data for the SL model by having human air traffic controllers solve the conflict scenarios generated in step 5. The State and Event logs of these simulations are stored. The State and Event logs are used to generate training input for the SL model, which consists of a picture of the Solution-Space-Diagram (SSD), which can be reconstructed form the State-log, and the action that was taken by the human air traffic controller to solve the conflict, which can be extracted from the Event-log. The SL model is then trained with the SSD and action data.
7	The SectorX ATC simulator is used to generate a small number of hand-crafted scenarios that will be presented to the human air traffic controller together with the resolution provided with the SL agent. These scenarios are then presented to the trained SL agent. This trained SL agent then provides resolutions/advisories, which are stored.

8	The stored resolutions/advisories are converted into SectorX event-logs. These event-logs can used, together with the original SectorX scenarios from step 7, to playback the scenario in SectorX and to present the resolutions/advisories that were generated by the SL agent to the human air traffic controller. Several options for providing transparency of the generated resolutions/advisories will be provided. The human air traffic controller can evaluate the proposed resolutions/advisories and accept or reject them. Also feedback can be given on the reason of acceptance/rejection.

The main effort in WP5 has been to set up this pipeline for testing of the integrated system. During testing, interfaces were identified that had to convert or transfer data from one component to the next, sometimes across different pieces of software and hardware. With different partners from the MAHALO project working on the integrated system, it was especially important to have clear interface definitions, since data sometimes had to be transferred between partners from different countries, with different programming styles and customs. The following chapter describes these interfaces.

3 Interface descriptions

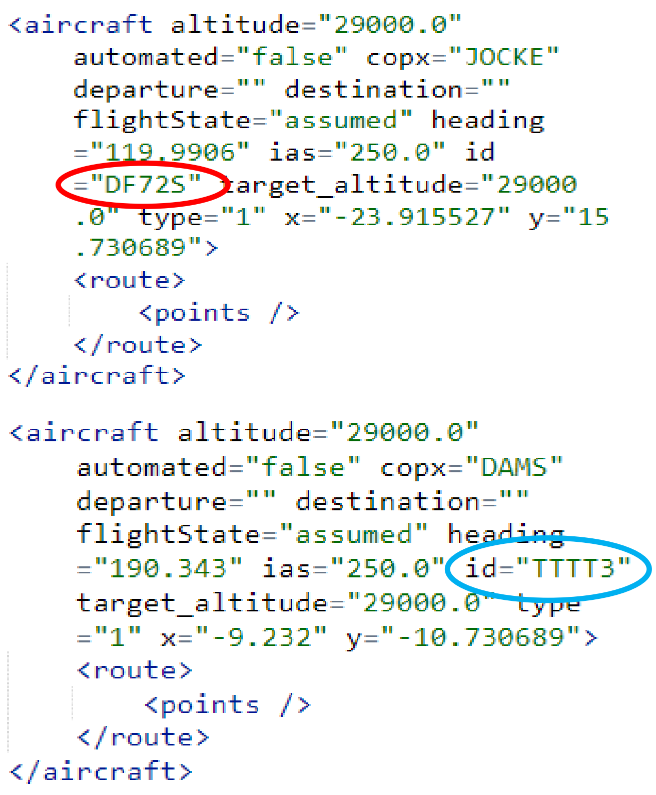
This chapter will describe the interfaces in the integrated system, where the machine learning models are presented in conjunction with the rest of the contextual information. First the Scenario description and interfaces will be presented in Section 3.1. The State and Event logs format will be presented in Section 3.2. Finally, the machine learning model interfaces will be presented in Section 3.3.

3.1 Scenario descriptions

The abstract concept of a scenario, in which a number of aircraft are situated in a sector, with certain position, velocity, heading information, has to be formalised and the format for storing scenarios must be defined.

3.1.1 Information stored in a scenario

A scenario file contains the basic information that is needed to initialize a simulation. It has information on the sector layout, waypoints, airports, routes, aircraft, and weather. In SectorX, and in our experiment, the airspace is 2-dimensional (x- and y-axis) with the origin being in the middle of the screen. Sectors can be as large as possible and any shape by defining end connecting end points. To run a scenario, a scenario file places the starting point of every aircraft with the angle at which the aircrafts are facing towards. A waypoint is specified with each aircraft that shows where the aircraft will leave the sector.



```
<aircraft altitude="29000.0"
    automated="false" copx="JOCKE"
    departure="" destination=""
    flightState="assumed" heading
    ="119.9906" ias="250.0" id
    ="DF72S" target_altitude="29000
    .0" type="1" x="-23.915527" y="15
    .730689">
    <route>
        <points />
    </route>
</aircraft>

<aircraft altitude="29000.0"
    automated="false" copx="DAMS"
    departure="" destination=""
    flightState="assumed" heading
    ="190.343" ias="250.0" id="TTTT3"
    target_altitude="29000.0" type
    ="1" x="-9.232" y="-10.730689">
    <route>
        <points />
    </route>
</aircraft>
```

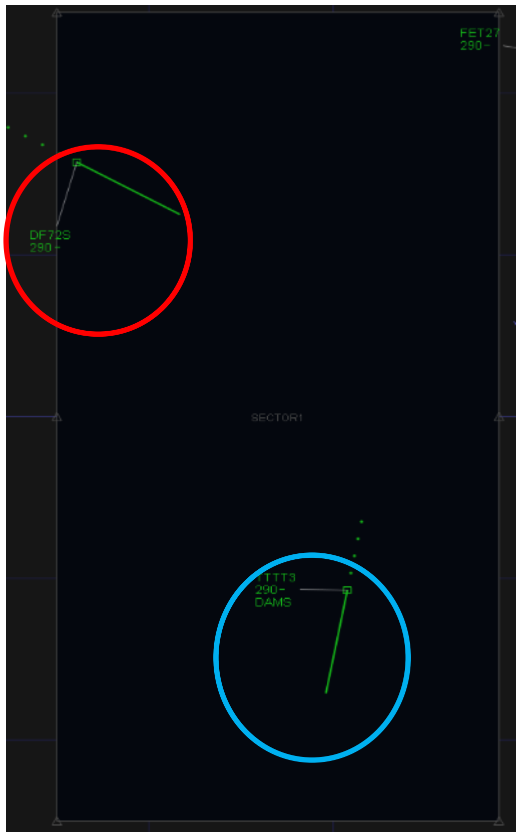


Figure 5: Traffic as expressed in a scenario file and displayed in SectorX

The specific format of a scenario file allows us to manipulate different aspects using scripts or manually to create different types of traffic scenarios. In these scenarios, some variables can be kept constant while others vary. Another feature worth mentioning is the automation feature on each aircraft. If this feature is enabled (Boolean variable set true) it allows specific aircraft to use a pre-built automation algorithm for detecting and avoiding conflicts with other aircraft.

3.1.2 Format of scenario file SectorX

Scenario files in the SectorX simulation environment are stored as XML files. Data format details for a SectorX scenario file can been found in Table 2. An example of a SectorX scenario file can be seen in Figure 5. An example of the specific scenarios used in SIM1 will be presented in deliverable D5.2 [3].

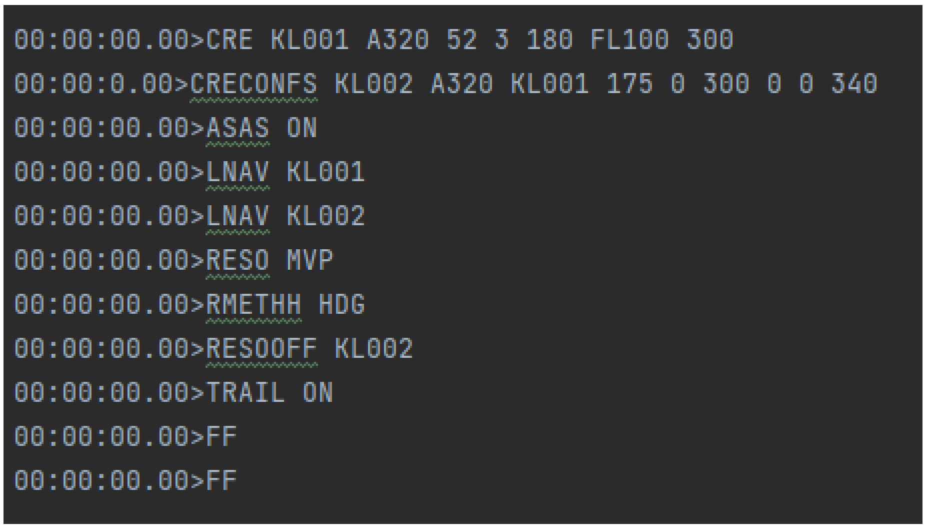
Table 2: SectorX scenario file format

Field	items	attributes	example		notes
root=scenario		rotation	0.0		degrees
airspace	sectors	sector points			Sector layout
	airports	-			
	waypoints	waypoint info			
	routes	-			
traffic	**aircraft**	id	FET27		callsign
		icao	A321		aircraft type
		x	0.17126441		x-position in NM w.r.t. center of sector
		y	-17.547169		y-position in NM w.r.t. center of sector
		heading	359.8956		degrees
		ias	250.0		indicated airspeed in knots
		altitude	29000.0		altitude in feet
		departure			
		destination			
		copx	AKON		target sector exit
		target_altitude	29000.0		traget altitude in feet
		exit_altitude_ft	29000.0		required altitude at sector exit in feet

		automated	false		boolean state for automation
		flightState	assumed		
weather	windfield	id	actual		
		type	empty		

3.1.3 Format of scenario file Bluesky

An example of a typical BlueSky scenario file can be seen below. This file follows a .scn format that can be readily edited through a notepad editor.



```
00:00:00.00>CRE KL001 A320 52 3 180 FL100 300
00:00:0.00>CRECONFS KL002 A320 KL001 175 0 300 0 0 340
00:00:00.00>ASAS ON
00:00:00.00>LNAV KL001
00:00:00.00>LNAV KL002
00:00:00.00>RESO MVP
00:00:00.00>RMETHH HDG
00:00:00.00>RESOOFF KL002
00:00:00.00>TRAIL ON
00:00:00.00>FF
00:00:00.00>FF
```

Figure 6: Example of a BlueSky scenario file

The repeating "00:00:00.00" string refers to the time at which each of these commands is to be carried out by BlueSky (in this case at *simtime* = 00:00:00.00, the start of the simulation). The main commands displayed above are summarized and explained in Table 3.

Table 3 Description of .scn file commands

Command in .scn	Functionality	Format
CRE	Creates an aircraft (ACID) in the simulation environment	CRE, ACID, TYPE, LAT, LONG, HDG, ALT, SPD
CRECONFS	Creates an aircraft (id) in conflict with the target aircraft (targeted)	CRECONFS, ID, TYPE, TARGETID, DPSI, CPA, TLOS_HOR, dH, TLOS_VER, SPD
ASAS	Switches Airborne Separation Assurance On/Off	ASAS ON/OFF

LNAV	Used to switch Lateral Navigation On/Off for a given aircraft	LNAV, ACID, SWITCH
RESO	Sets a resolution method (OFF, MVP, EBY, SWARM)	RESO MVP/OFF (the other methods are not used in this research)
RMETHH	Sets horizontal resolution using speed (SPD) or heading (HDG)	RMETHH BOTH/SPD/HDG/NONE
RESOOFF	Switches off conflict resolution module for given aircraft (Used to disable intruding aircraft CR)	RESOOF ACID
TRAIL	Used to switch on/off visual trails of the aircraft	TRAIL OFF/ON
FF	Fast forward the simulation to conduct it faster than real time	FF

Let us now analyse the above example for the .scn file. The .scn file in question is initializing two A320s. The first one (controlled a/c) is initialized through an absolute latitude and longitude, the second one (intruding a/c) is initialized such that there will be a conflict with a conflict angle of 175 degrees and a d_{cpa} of 0nms with a time to loss of separation of 300seconds. ASAS is then turned on, the resolution method is selected as MVP (with only horizontal, heading resolutions) and the resolutions for aircraft 2 are turned off so that only the controlled a/c carries out resolution commands. The trails are then turned on for better visual inspection and the simulation is fast-forwarded.

3.1.4 Conversion of scenario files from SectorX to Bluesky

The conversion of scenario files from SectorX to BlueSky is performed by a *python* script. This script extracts all necessary information from the SectorX log and outputs it into BlueSky form. There are some pieces of information that are contained in the SectorX logs that are not put into the .scn file for BlueSky. These have to do with the sector layout itself. The .scn file from BlueSky does not directly handle sector definition, it is mostly focused on initializing the aircraft, collision avoidance and plugins (which are used to add further functionality to BlueSky). This means that, while this information is contained in the SectorX scenario file, it is not directly extracted and is instead manually added to the BlueSky files corresponding to sector definition. This is not a problem since the scenarios tend to differ in their traffic and choice of controlled aircraft but not in their geography or location. For the purposes of MAHALO the sector layouts and locations were kept constant to control for sector variation having an effect on the measurements. This means that the sector information only needed to be manually added once. Therefore, the scenario converter can automatically grab all information about the aircraft and place them into a scenario file, knowing that the sector is already correctly defined.

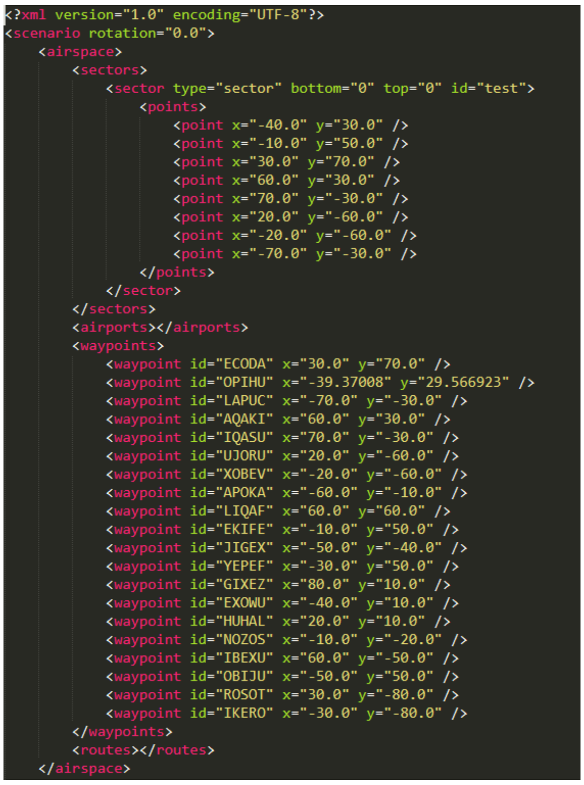
Since the SectorX scenario files are in .xml format, the library *xml.etree.ElementTree* is used to parse these .xml files and get the information required. By default the converter will analyse and convert every file that is in the same folder as it is and corresponds to the proper SectorX format into the

BlueSky .scn format. This is done so that scenarios can be quickly converted in batches by simply putting them in the same folder as the scenario converter and running it once.

Comparing Figure 6 with Figures 7 and 8 one can see that the aircraft information can be readily transferred from one type of scenario file to the other. With one exception, the positions in SectorX are defined in a local XYZ reference frame whereas BlueSky uses latitude/longitude/altitude. A conversion is used to convert these XYZ coordinates into approximate Lat/Long coordinates. The conversion used is:

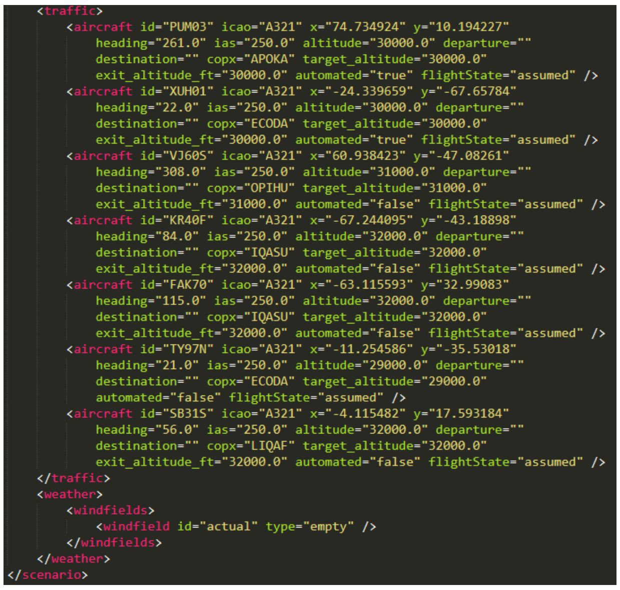
$$1\,\mathrm{min}\,latitude = 1nm; \qquad 1\,\mathrm{min}\,longitude = \cos(lat)\,nm$$

This conversion does carry some error but at the distances considered this error was deemed to not be overly significant. One should note that this conversion becomes worse the closer to the poles the sector is considered. Other more accurate conversions can also be used if a higher level of detail is desirable. For the purposes of MAHALO this was deemed acceptable.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<scenario rotation="0.0">
    <airspace>
        <sectors>
            <sector type="sector" bottom="0" top="0" id="test">
                <points>
                    <point x="-40.0" y="30.0" />
                    <point x="-10.0" y="50.0" />
                    <point x="30.0" y="70.0" />
                    <point x="60.0" y="30.0" />
                    <point x="70.0" y="-30.0" />
                    <point x="20.0" y="-60.0" />
                    <point x="-20.0" y="-60.0" />
                    <point x="-70.0" y="-30.0" />
                </points>
            </sector>
        </sectors>
        <airports></airports>
        <waypoints>
            <waypoint id="ECODA" x="30.0" y="70.0" />
            <waypoint id="OPIHU" x="-39.37008" y="29.566923" />
            <waypoint id="LAPUC" x="-70.0" y="-30.0" />
            <waypoint id="AQAKI" x="60.0" y="30.0" />
            <waypoint id="IQASU" x="70.0" y="-30.0" />
            <waypoint id="UJORU" x="20.0" y="-60.0" />
            <waypoint id="XOBEV" x="-20.0" y="-60.0" />
            <waypoint id="APOKA" x="-60.0" y="-10.0" />
            <waypoint id="LIQAF" x="60.0" y="60.0" />
            <waypoint id="EKIFE" x="-10.0" y="50.0" />
            <waypoint id="JIGEX" x="-50.0" y="-40.0" />
            <waypoint id="YEPEF" x="-30.0" y="50.0" />
            <waypoint id="GIXEZ" x="80.0" y="10.0" />
            <waypoint id="EXOWU" x="-40.0" y="10.0" />
            <waypoint id="HUHAL" x="20.0" y="10.0" />
            <waypoint id="NOZOS" x="-10.0" y="-20.0" />
            <waypoint id="IBEXU" x="60.0" y="-50.0" />
            <waypoint id="OBIJU" x="-50.0" y="50.0" />
            <waypoint id="ROSOT" x="30.0" y="-80.0" />
            <waypoint id="IKERO" x="-30.0" y="-80.0" />
        </waypoints>
        <routes></routes>
    </airspace>
```

Figure 7: Example of SectorX scenario file, part 1



```
    <traffic>
        <aircraft id="PUM03" icao="A321" x="74.734924" y="10.194227"
            heading="261.0" ias="250.0" altitude="30000.0" departure=""
            destination="" copx="APOKA" target_altitude="30000.0"
            exit_altitude_ft="30000.0" automated="true" flightState="assumed" />
        <aircraft id="XUH01" icao="A321" x="-24.339659" y="-67.65784"
            heading="22.0" ias="250.0" altitude="30000.0" departure=""
            destination="" copx="ECODA" target_altitude="30000.0"
            exit_altitude_ft="30000.0" automated="true" flightState="assumed" />
        <aircraft id="VJ60S" icao="A321" x="60.938423" y="-47.08261"
            heading="308.0" ias="250.0" altitude="31000.0" departure=""
            destination="" copx="OPIHU" target_altitude="31000.0"
            exit_altitude_ft="31000.0" automated="false" flightState="assumed" />
        <aircraft id="KR40F" icao="A321" x="-67.244095" y="-43.18898"
            heading="84.0" ias="250.0" altitude="32000.0" departure=""
            destination="" copx="IQASU" target_altitude="32000.0"
            exit_altitude_ft="32000.0" automated="false" flightState="assumed" />
        <aircraft id="FAK70" icao="A321" x="-63.115593" y="32.99083"
            heading="115.0" ias="250.0" altitude="32000.0" departure=""
            destination="" copx="IQASU" target_altitude="32000.0"
            exit_altitude_ft="32000.0" automated="false" flightState="assumed" />
        <aircraft id="TY97N" icao="A321" x="-11.254586" y="-35.53018"
            heading="21.0" ias="250.0" altitude="29000.0" departure=""
            destination="" copx="ECODA" target_altitude="29000.0"
            automated="false" flightState="assumed" />
        <aircraft id="SB31S" icao="A321" x="-4.115482" y="17.593184"
            heading="56.0" ias="250.0" altitude="32000.0" departure=""
            destination="" copx="LIQAF" target_altitude="32000.0"
            exit_altitude_ft="32000.0" automated="false" flightState="assumed" />
    </traffic>
    <weather>
        <windfields>
            <windfield id="actual" type="empty" />
        </windfields>
    </weather>
</scenario>
```

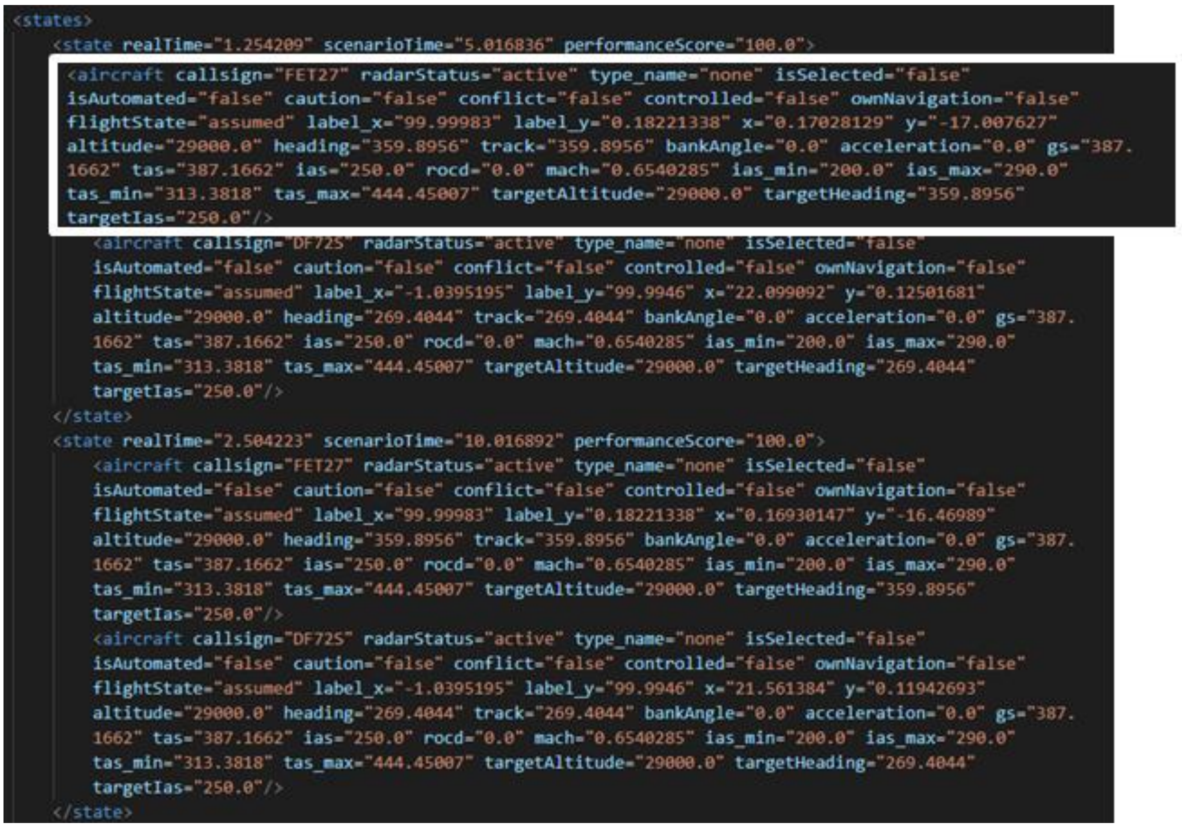
Figure 8: Example of SectorX scenario file, part 2 (continued from part 1)

3.2 States and Events

For the conformal ML model shown in Figure 4 state and event information, generated in the SectorX during the prequel Human-in-the-loop tests, has to be processed such that it can be used for training of the SL model. All information from the ATC simulation and the human interaction with it is stored in two types of files: SectorX State-logs and SectorX Event-logs

3.2.1 SectorX State-logs

In SectorX, the State-logs are used to capture the state of the whole ATC simulation. At fixed rate in the simulation, a snapshot is taken from the current state of the simulation. This information is stored in an XML file with a fixed structure. An example of a SectorX state-log file can be seen in Figure 9. Most information in the state-log is related to the states of the aircraft in the simulation, but there is also information on the simulation timestamp and the scenario timestamp. Table 4 shows the SectorX state-log definition together with example values, corresponding to Figure 9.



```
<states>
    <state realTime="1.254209" scenarioTime="5.016836" performanceScore="100.0">
    <aircraft callsign="FET27" radarStatus="active" type_name="none" isSelected="false"
    isAutomated="false" caution="false" conflict="false" controlled="false" ownNavigation="false"
    flightState="assumed" label_x="99.99983" label_y="0.18221338" x="0.17028129" y="-17.007627"
    altitude="29000.0" heading="359.8956" track="359.8956" bankAngle="0.0" acceleration="0.0" gs="387.
    1662" tas="387.1662" ias="250.0" rocd="0.0" mach="0.6540285" ias_min="200.0" ias_max="290.0"
    tas_min="313.3818" tas_max="444.45007" targetAltitude="29000.0" targetHeading="359.8956"
    targetIas="250.0"/>
        <aircraft callsign="DF725" radarStatus="active" type_name="none" isSelected="false"
        isAutomated="false" caution="false" conflict="false" controlled="false" ownNavigation="false"
        flightState="assumed" label_x="-1.0395195" label_y="99.9946" x="22.099092" y="0.12501681"
        altitude="29000.0" heading="269.4044" track="269.4044" bankAngle="0.0" acceleration="0.0" gs="387.
        1662" tas="387.1662" ias="250.0" rocd="0.0" mach="0.6540285" ias_min="200.0" ias_max="290.0"
        tas_min="313.3818" tas_max="444.45007" targetAltitude="29000.0" targetHeading="269.4044"
        targetIas="250.0"/>
    </state>
    <state realTime="2.504223" scenarioTime="10.016892" performanceScore="100.0">
        <aircraft callsign="FET27" radarStatus="active" type_name="none" isSelected="false"
        isAutomated="false" caution="false" conflict="false" controlled="false" ownNavigation="false"
        flightState="assumed" label_x="99.99983" label_y="0.18221338" x="0.16930147" y="-16.46989"
        altitude="29000.0" heading="359.8956" track="359.8956" bankAngle="0.0" acceleration="0.0" gs="387.
        1662" tas="387.1662" ias="250.0" rocd="0.0" mach="0.6540285" ias_min="200.0" ias_max="290.0"
        tas_min="313.3818" tas_max="444.45007" targetAltitude="29000.0" targetHeading="359.8956"
        targetIas="250.0"/>
        <aircraft callsign="DF725" radarStatus="active" type_name="none" isSelected="false"
        isAutomated="false" caution="false" conflict="false" controlled="false" ownNavigation="false"
        flightState="assumed" label_x="-1.0395195" label_y="99.9946" x="21.561384" y="0.11942693"
        altitude="29000.0" heading="269.4044" track="269.4044" bankAngle="0.0" acceleration="0.0" gs="387.
        1662" tas="387.1662" ias="250.0" rocd="0.0" mach="0.6540285" ias_min="200.0" ias_max="290.0"
        tas_min="313.3818" tas_max="444.45007" targetAltitude="29000.0" targetHeading="269.4044"
        targetIas="250.0"/>
    </state>
```

Figure 9: Example of a SectorX State-log

Table 4: SectorX State-log definition (example values correspond to Figure)

Top-level variable	Subfield variable	Example value		Description
realTime		1.254209		Time since start of simulation in seconds
scenarioTime		5.016836		Time inside the simulation in seconds
performanceScore		100		Can be used to store a performance measure (to be defined)
aircraft	callsign	FET27		callsign/ aircraft ID
	radarStatus	active		indicates if aircraft is currently active on the radar display
	type_name	none		aircraft type
	isSelected	false		indicates if aircraft is currently selected on the radar display
	isAutomated	false		indicates whether aircraft is under automated control
	caution	false		indicates if there is a caution for the given aircraft

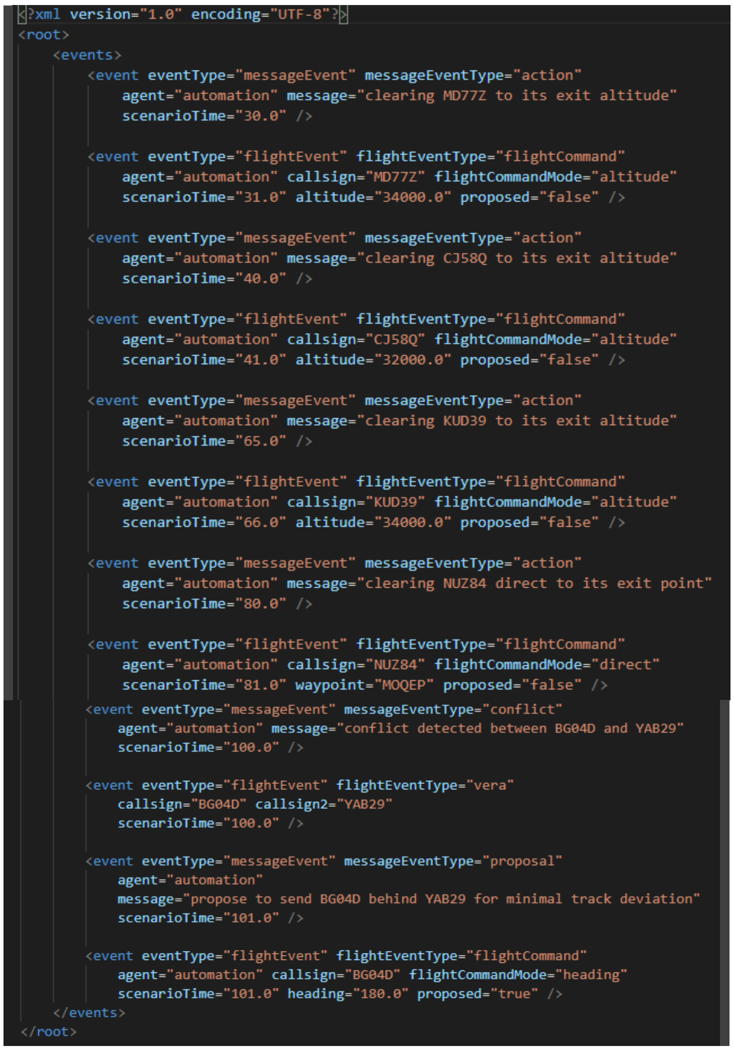
	conflict	false		indicates if there is a conflict for the given aircraft
	controlled	false		indicates if aircraft is currently being controlled
	ownNavigation	false		indicates if aircraft is put under own navigation
	flightState	assumed		current state of the flight
	label_x	99.99983		position of aircraft label in x-direction
	label_y	0.18221338		position of aircraft label in y-direction
	x	0.17126441		x-position in NM
	y	-17.547169		y-position in NM
	altitude	29000.0		feet
	heading	359.8956		heading angle in degrees
	track	359.8956		track angle in degrees (equal to heading angle if no wind)
	bankAngle	0.0		bank angle in degrees
	acceleration	0.0		aircraft acceleration
	gs	387.1662		ground speed in knots (same as tas if no wind)
	tas	387.1662		true airspeed in knots
	ias	250.0		indicated airspeed in knots
	rocd	0		rate of climb/descent in feet/min
	mach	0.6540285		Mach number
	ias_min	200.0		minimum allowed indicated airspeed in knots
	ias_max	290.0		maximumallowed indicated airspeed in knots
	tas_min	313.3818		minimum allowed true airspeed in knots
	tas_max	444.45007		maximumallowed true airspeed in knots
	targetAltitude	29000.0		target altitude in feet
	targetHeading	359.8956		target heading in degrees
	targetIas	250.0		target inidcated airspeed in knots

3.2.2 SectorX Event-logs

SectorX Event-logs record any interaction that is made with the simulation, either by human operators, or by the machine learning models. Human interaction with the simulator includes commands that are given to aircraft, such as a requested heading or altitude change, but also includes all interactions that are made through the human-machine-interface, such as mouse clicks, menu button clicks, etc. The machine learning interaction is purely through provided resolutions, no other event types, such as mouse clicks are performed by the machine learning models.

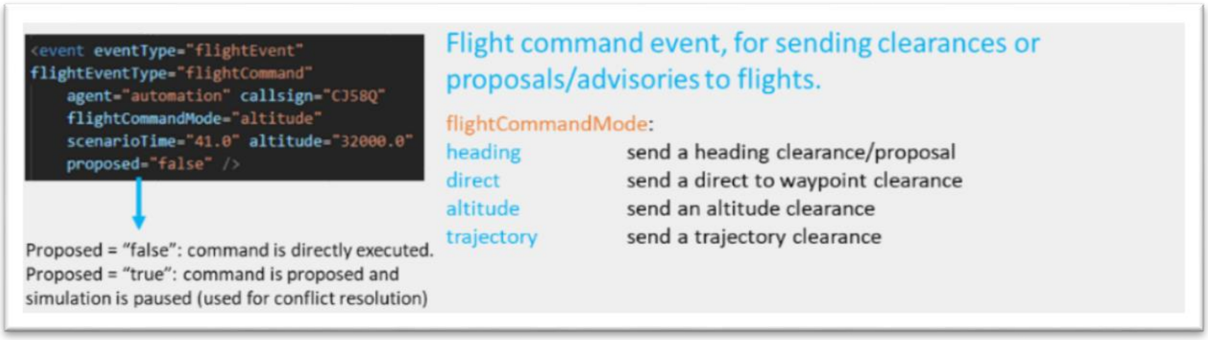
Figure 10 shows an example of a SectorX Event-log. Events have specific types, such as "messageEvent" which means a message was shown on the display, or a "flightEvent," which can be a command given by automation or by the human operator to an aircraft that is controlled in the sector.

Figure 11 shows an explanation of what can be contained in a "flightEvent." The specific subfields in the Event-log are overlapping with some subfields in the State-log, so Table 4 can be used again to see a description of these variables.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <events>
        <event eventType="messageEvent" messageEventType="action"
            agent="automation" message="clearing MD77Z to its exit altitude"
            scenarioTime="30.0" />

        <event eventType="flightEvent" flightEventType="flightCommand"
            agent="automation" callsign="MD77Z" flightCommandMode="altitude"
            scenarioTime="31.0" altitude="34000.0" proposed="false" />

        <event eventType="messageEvent" messageEventType="action"
            agent="automation" message="clearing CJ58Q to its exit altitude"
            scenarioTime="40.0" />

        <event eventType="flightEvent" flightEventType="flightCommand"
            agent="automation" callsign="CJ58Q" flightCommandMode="altitude"
            scenarioTime="41.0" altitude="32000.0" proposed="false" />

        <event eventType="messageEvent" messageEventType="action"
            agent="automation" message="clearing KUD39 to its exit altitude"
            scenarioTime="65.0" />

        <event eventType="flightEvent" flightEventType="flightCommand"
            agent="automation" callsign="KUD39" flightCommandMode="altitude"
            scenarioTime="66.0" altitude="34000.0" proposed="false" />

        <event eventType="messageEvent" messageEventType="action"
            agent="automation" message="clearing NUZ84 direct to its exit point"
            scenarioTime="80.0" />

        <event eventType="flightEvent" flightEventType="flightCommand"
            agent="automation" callsign="NUZ84" flightCommandMode="direct"
            scenarioTime="81.0" waypoint="MOQEP" proposed="false" />
        <event eventType="messageEvent" messageEventType="conflict"
            agent="automation" message="conflict detected between BG04D and YAB29"
            scenarioTime="100.0" />

        <event eventType="flightEvent" flightEventType="vera"
            callsign="BG04D" callsign2="YAB29"
            scenarioTime="100.0" />

        <event eventType="messageEvent" messageEventType="proposal"
            agent="automation"
            message="propose to send BG04D behind YAB29 for minimal track deviation"
            scenarioTime="101.0" />

        <event eventType="flightEvent" flightEventType="flightCommand"
            agent="automation" callsign="BG04D" flightCommandMode="heading"
            scenarioTime="101.0" heading="180.0" proposed="true" />
    </events>
</root>
```

Figure 10: SectorX Event-log example



```
<event eventType="flightEvent"
flightEventType="flightCommand"
    agent="automation" callsign="CJ58Q"
    flightCommandMode="altitude"
    scenarioTime="41.0" altitude="32000.0"
    proposed="false" />
```

Proposed = “false”: command is directly executed.
Proposed = “true”: command is proposed and simulation is paused (used for conflict resolution)

Flight command event, for sending clearances or proposals/advisories to flights.

flightCommandMode:

heading	send a heading clearance/proposal
direct	send a direct to waypoint clearance
altitude	send an altitude clearance
trajectory	send a trajectory clearance

Figure 11: SectorX Event type information

With the Scenario definition, State and Event-logs definitions, and the conversions that have to be made between the Bluesky and SectorX simulators in place, the remaining part of the interface is a description of the machine learning models input and outputs. These inputs/outputs connect to State and Event logs to form an integrated system. The following section will describe these ML interfaces in detail.

3.3 Machine Learning model description

The main working principle of the ML models used in the MAHALO project can been found in deliverable D3.1 Machine Leaning report [1]. In this section, the focus is on the interaction of the ML models with the rest of the system, i.e. how data is pre-processed and fed to the ML model and how is the output of the ML models used to provide the suggested resolution to the user.

3.3.1 Conformal SL model

The main task of the SL model is to act analogously to the ATCO. This entails seeing the same information that the ATCO sees on the display of SectorX and generating conflict resolution actions that reproduce the ATCO’s actions for similar conflicts. Technically, this can be done in several ways. As mentioned, we employ the Solution Space Diagram (SSD) to observe the surroundings of the aircraft the ATCO is focusing on. This is the input for the SL model in a macro sense, for both training and practically, when using the agent. Figure 12 shows an example of an instance of a conflict and how the SSD appears on SectorX from the ATCO’s view. For more details on the pipeline for the SL model and how images are generated for training the model, see Chapter 3 in Deliverable 5.2 Simulation 1 report [3].

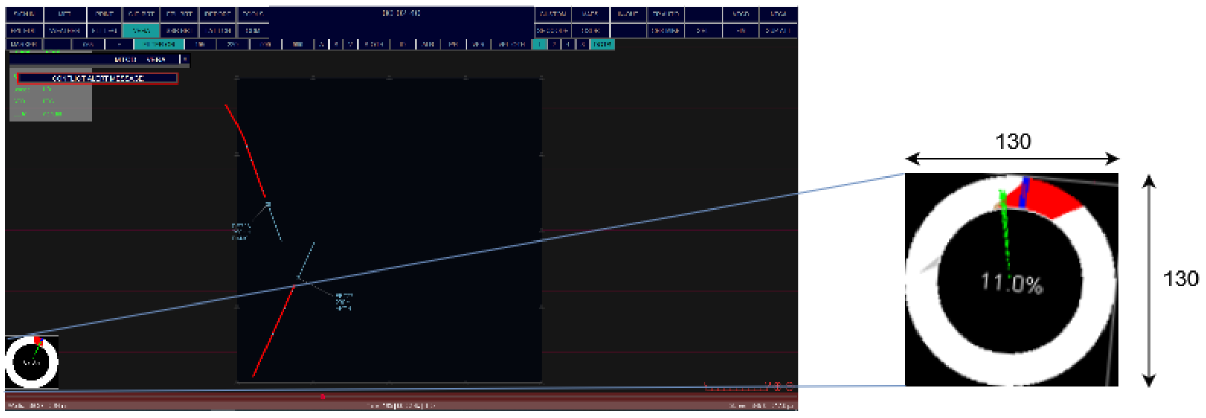


Figure 12: SSD image as input to the SL model

The SSD images generated are of equal 130 pixel per inch size. The pixels are handled as a matrix (i.e., a NumPy array), pre-processed by resizing if necessary, normalized as values between 0 and 1, and eventually labelled by matrix. The label represents the heading change that the ATCO made during an instance to avoid the conflict between aircrafts. When trained using conflict resolutions from individual ATCOs, this data allows the conformal SL agent to know and model, for every instance of the SSD, specific air controller decisions-- for example, a decision on Aircraft 1 to make a 20 degrees turn west.

For efficient and smooth training, labels are pre-processed before training for simplification. Since the heading changes could be any numerical (i.e., float) between 0 and 360, this could complicate classification and more data would be needed for a higher confidence. The agent does not have to be very specific with a heading suggestion having multiple decimal points. Hence, decisions are simplified and categorized in 4 different states using SSD images with either +20, +10, -10 or -20 heading angle changes. This is the approach being taken as of now and results in angle predictions by the model as a list containing 4 values. These values sum to 1, and the one with the maximum value indicates the model suggestion and heading change for the specific instance (SSD image). Subsequently, this result can be used to further enhance the output from the agent. For example, since the SL model pipeline already has the information of the specific aircraft and its corresponding SSD image, it would be possible to display detailed information to the ATCO in SectorX. The pipeline as constructed now, returns the '*Callsign'* (aircraft) and '*Scenario Time'* in addition to the '*Heading change'* as results from the SL agent.

To reiterate, the agent being compiled using TensorFlow in Python takes an array of pixels, preprocesses it, and assigns a label (representing the decision). A more detailed technical description of the supervised learning topology used for the SL model can be found in MAHALO deliverable D3.1.

3.3.2 Optimal RL model

The RL agents require only information on the states of the different aircraft in the sector, as their goal is not to be conformal to the operator. For the two different RL agents, the Q-learning and DQfD agents, the inputs and outputs differ.

3.3.2.1 Inputs

When it comes to the Q-learning agent, since it only acts as a "supervisor" to manage and oversee the performance of the MVP algorithm, its inputs are relatively simple. The Q-learning agent does not have direct control on the commands given to the aircraft and, instead, controls the parameters of the MVP algorithm. Therefore, the inputs of the Q-learning agent are:

- Conflict geometry parameters: conflict angle, initial dcpa (distance to closest point of approach at the start of the scenario) and initial time to loss of separation (tlos in seconds) of the two aircraft
- Previous settings of the MVP algorithm: lookahead time and safety factor

These inputs are also the states of the Q-learning agent.

The DQfD agent directly provides a resolution, unlike the Q-learning model, which uses the MVP algorithm as an intermediate. The DQfD agent then receives information on the current traffic state through pixel information. This pixel information is codified in the aforementioned SSD. In order to increase accuracy, DQfD is fed two SSDs corresponding to two successive states. This provides the agent with more contextual information

3.3.2.2 Outputs

The Q-learning agent decides the parameters the MVP algorithm is to use. Therefore its outputs are made up of the two parameters it is trained to optimize: lookahead time and safety factor. These are outputted as numerical values.

The DQfD agent outputs a combined heading + speed solution to the controlled aircraft. This is given as a numerical value as well. For better visualization it can then be converted back into an SSD (with the command represented as an arrow), but this is only relevant for the interaction with the operator.

4 Conclusions and Recommendations

This chapter presents the main conclusions and outcomes of the integration process in WP5, with a summary of achieved results and lessons learned.

An integration pipeline for combining the Human Machine Interface and the Machine Learning models was successfully designed and implemented according to the diagrams in Figure 3 and Figure 4. Verification of the scripts that allow a transformation of data from one simulator into the other has been carried out by visual inspection of the generated simulation in both simulators and the results were found to be good.

The real test to see if all integration steps have been performed correctly is performed in the SIM1 experiment. MAHALO Deliverable 5.2 [3] describes SIM1 and its outcomes in detail, so that will not be repeated here, but it can be mentioned here that the pipeline is working as intended and data flows correctly from human operators to the simulator, from simulators to the machine learning models and also back to the human operators again.

A challenge in designing the integration process and in defining the interfaces was that not all steps had been fully worked out yet. For example, there was still work being done on creating the right type of scenario for the Main Experiment [5]. Depending on the design choice of the scenario, the scenario interface had to be updated. Another example is in the type of resolutions that the ML models have to provide. If the resolution is only in heading, then a “heading Command” Event-type suffices. If however, the ML models also have to generate a speed or altitude command, then the Event-type definition has to be updated.

Ideally all subcomponents should already be designed and frozen when the integration takes place, but, on the other hand, the integration process itself sometimes identifies issues in the components that have to be resolved, either by manipulating the interface, or by changing the internals of the component. In the end this will also improve the complete integrated system.

A key lesson learned is that the interface design should be started early, such that missing functionality in the components can be found early as well. Also all interface definitions should be continually updated and shared between all members working on the components. This sounds very obvious, but in practice it has occurred, also in this project, that component inputs/outputs were updated without changing the interface definition, which can result in an incorrectly operating system.

A specific component output that was found to be missing during interface design is the way in which the ML models will provide explanation about their provided resolution. The format of the output has been defined, e.g. a 200 character string of text, but it became clear that what exactly we want to put in that text was not designed yet. This identified problem was addressed in the MAHALO workshop held on October 28, 2021, in which we asked the participants (among which experienced Air Traffic Controllers) what kind of explanation they would like to receive form the ML models. The outcome of this workshop is documented in MAHALO deliverable 7.2[4] and will be used for the design of the final experiments in MAHALO WP6.

5 References

[1] MAHALO D3.1 Machine Learning report

[2] MAHALO D 4.1 E-UI design doc & demonstrator

[3] MAHALO D5.2 Simulation 1 report

[4] MAHALO D7.2 Workshop report

[5] MAHALO D6.1 Experimental design document