TREBALL DE FI DE CARRERA

Title: eDEP integration and validation in the ISIS simulation platform

Author: Carlos Martí Cudinach

Director: Pablo Royo Chic.

Marc Perez Batlle.

Date: July, 21th 2011

Title: eDEP integration and validation in the ISIS simulation platform

Author: Carlos Martí Cudinach

Director: Pablo Royo Chic.

Marc Perez Batlle.

Date: July, 21th 2011

Overview

The Intelligent Communications and Avionics for Robust Unmanned Aerial Systems group (ICARUS group) are formed by researchers of the Technical University of Catalonia. One of research lines is the automation and development of on-board avionics and ground systems in order to support either manned and Unmanned Aircraft Systems (UAS) while providing high levels of flexibility and low development costs.
The ICARUS group developed a UAS simulation platform called ISIS which simulates UAS missions. This platform allows to study their behavior in interaction with the environment. However, it soon appeared the need to simulate more complex operations in which the UAS interacted not only the environment but also with air traffic.
Eurocontrol (European Organisation for the Safety of Air Navigation) provided an Air Traffic Management (ATM) simulator platform called eDEP to ICARUS group. It opened the possibility of developing a simulation tool for integrating UAS into non-segregated airspace.
This project has consisted of integrating these two simulation tools to build a new version of the UAS simulator able to integrate air traffic on ISIS platform. Air traffic will allow to create simulations in more complex environments.

ÍNDEX

INTRODUCTION

The Intelligent Communications and Avionics for Robust Unmanned Aerial Systems group (ICARUS group) are formed by researchers of the Technical University of Catalonia. One of research lines is the automation and development of on-board avionics and ground systems in order to support either manned and Unmanned Aerial Systems (UAS) while providing high levels of flexibility and low development costs.

Unmanned Aircraft Systems (UAS) are becoming one of the main assets to be employed in remote sensing applications both in civil and scientific environments. However, the current limitations when using non-segregated airspace makes it extremely difficult to have available flight time to extensively test all subsystems required to achieve a specific mission.

Using real flights to test the complete UAS mission infrastructure involves some costs and risks. Current legislation requires an area of segregated airspace to perform the test flights. Also, except for the smaller UAS, many personnel involved in the UAS mission also need to be mobilized. In addition, government permissions and adequate insurance coverage for the UAS operation has to be obtained. Finally, the possibility of damaging or completely losing the UAS should be considered, especially when testing new components or subsystems. Therefore, simulation is a pressing requirement prior to real flight campaigns [1].

The ICARUS group developed an UAS simulation platform which simulates UAS missions to solve these problems. This platform allows to study their behavior in interaction with the environment. However, it soon appeared the need to simulate more complex operations in which the UAS interacted not only the environment but also with air traffic.

Thanks to Eurocontrol (European Organisation for the Safety of Air Navigation) that provided an Air Traffic Management (ATM) simulator platform to ICARUS group. It opened the possibility of developing a simulation tool for integrating UAS into non-segregated airspace.

This project has consisted of integrating these two simulation tools to build a new version of the UAS simulator able to integrate air traffic on its platform. This integration will allow to create simulations in more complex environments. UAS simulated on ISIS will interact in non-segregated airspace with eDEP air traffic.

In addition, with this integration, the user will be able to include the UAS in a Visual Flight Rules (VFR) and Instrumental Flight Rules (IFR) scenarios designed on eDEP It will allow to study the simulated UAS on ISIS its behavior when interacts with air traffic.

Therefore the objectives of this project are:

- Homogenize data between the two simulation platforms.
- To group the air traffic from the ATM simulator (eDEP) to UAS simulator (ISIS) in a real time.

- Include the UAS in the air traffic simulator as an aircraft more.
- To Design and test simulations for a UAS interacting with air traffic.

This document is structured into six chapters as follows:

- In the Chapter 1 it explains the basic concepts for understanding the project and describes the two platforms involved in the integration.
- Chapter 2 shows the architecture of the project, how the modules programmed exchange data between the platforms. In addition, this chapter shows three versions or upgrades due to working time.
- Chapter 3 discusses how the project manages the data, how data flows between the two simulation platforms.
- Chapter 4 explains the implementation of the project, how did the architecture and design of the project and how it evolved over time.
- Chapter 5 shows the two tests of the integration between the two simulation platforms. The first simulation is in Visual Flight Rules (VFR) procedures and the second in Instrumental Flight Rules (IFR) procedures.
- Chapter 6 is referred to the conclusions and improvements of the project.

CHAPTER 1. PREVIOUS WORK

Before explaining this project is needed understanding of certain essential concepts in order to understand all the work done. That's why this chapter seeks to clarify these concepts explaining the UAS concepts and air traffic simulation platforms. In addition all the elements involved in the integration of the two simulators are described.

This chapter is structured in two parts. In the first part of the chapter we will see the UAS simulation platform (ISIS) and eDEP. In the second part we will explain the concepts involved in the project .This is, the ones that will be appearing in this document.

1.1. Simulation Platforms

As we have mentioned before, we have used UAS and ATM simulation platforms in this project. The platforms involved to be integrated are:

- ICARUS [1] Simulation Integrated Scenario (ISIS).
- early Demonstration & Evaluation Platform (eDEP).

1.1.1. ICARUS Simulation Integrated Scenario (ISIS)

In this section we explain the UAS simulation platform called Icarus Simulation Integrated Scenario (ISIS) [2]. During the explanation of this section we will see the need to create a simulation platform and a short summary of its performance. Finally, we will see a brief comment about my experience with the simulation platform.

All vehicles need a simulation to verify their behavior when they interacts with the environment. With this premise simulation, ensures a pattern of operation before starting the testing phase and allows to control potential errors. The UAS with testing do not escape this condition because they have a set of drawbacks. This drawbacks are describe below:

- The rules of the use of airspace allows to the UAS to only fly in segregated airspace.
- Use real flight to test, has a costs of fuel and infrastructure (mobilize the ground station) and a risk of accident.
- The weather conditions may affect the tests.

To solve this problems ICARUS group created “ICARUS Simulation Integrated Scenario” (ISIS) which is a civil UAS simulation platform. ISIS is a simulation platform which is able to simulate a UAS. Thanks to designed software components called “services” ISIS simulate the environment interacting with the

UAS accurately. Among many of these software components, we find the possibility of providing the UAS autopilot with a flight plan designed by the user.

The visualization and the flight dynamics of the simulation is performed through a flight simulator called Flight Gear[1]. The figure 1.1 the Shadow MK airframe model is illustrated inside the FlightGear simulator and the figure 1.2 shows his flight plan display [2]:

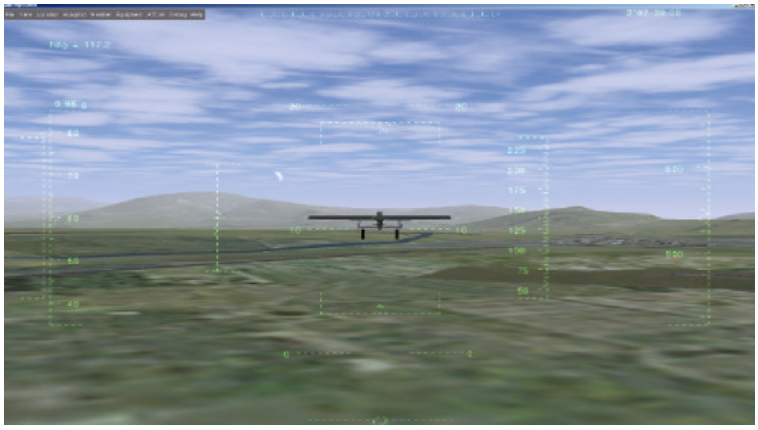

Figure 1.1. Flight gear screenshot of Shadow MK simulated with ISIS

Figure 1.2. Shadow MK flight plan tracking display

Below we will see in a summary way how ISIS works. This short summary will show us the organization of ISIS and how the software components interact with each other.

This simulator has an architecture that allows to provide an environment in which the designed software components called “Services” implement the required functionalities for the accomplishment of the UAS mission simulated.

[1] Flight Gear: The FlightGear flight simulator project is an open-source, multi-platform, cooperative flight simulator development project.

Services interact with each other using a Local Area Network (LAN) as a communicating system [4]. The Figure 1.3 illustrates the architecture of ISIS.

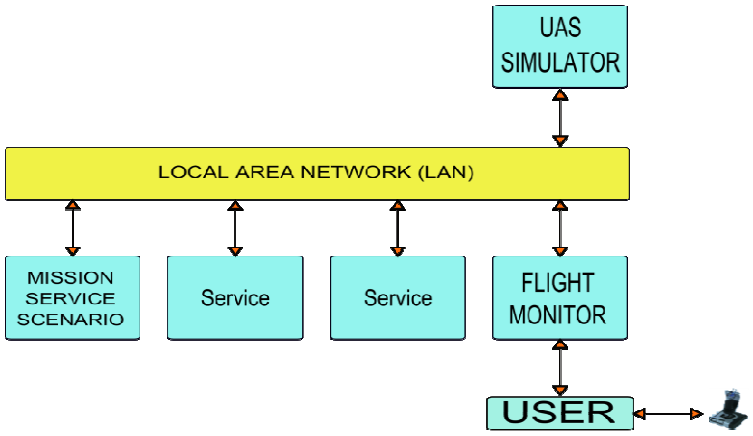

Figure 1.3. ISIS architecture

But the Local Area Network (LAN) needs a software system to communicate different services and manage them. For this reason, there is a software system called “Middleware Architecture for Remote Embedded Applications” (MAREA) over the LAN.

MAREA [4] is a middleware which provides an execution environment with communication channels and common functionalities. MAREA is a software system based on communication of different services in the local area network (LAN). It is responsible for starting and stopping all services. It also monitors their behavior right service. To do this, and in order to achieve rapid prototyping, managing a list of services beginning with the initial services.

Services do not access the network directly. All their communication is carried through the middleware. MAREA abstracts the network access, allowing the services to be deployed in different nodes depending on the UAS configuration.

The role of each service is expressed by the action of publishing, subscribing, or both simultaneously, thus, the model of publish / subscribe eliminates complex network programming for distributed applications that makes it easy to implement an integrated service. Marea offers the location of the other services and manages its discovery within the network, which controls all transfer tasks, message addressing and distribution, data delivery, flow control, etc.

Information exchange is carried out through four communication primitives, which have been named as Variables, Events, Remote Invocations and File Transmissions. For this project we use the Variables and Events primitives. So we explain two communication primitives:

- Variables: The variables are structured transmission of the information from one service to one or more services in the distributed system. A

variable can be sent at regular intervals or whenever there is a substantial change in value. The system must be able to tolerate the loss of one or more of these data transmissions. The primitive communication variable is the paradigm of subscription publication.

- Events: As the variables, the events also follow the paradigm of subscription publication. The main difference with respect to the variables is that events must ensure receipt of information sent signed by all services. The utility of events is to inform all interested in any casual and important fact. Some examples are the error messages or warnings, and directions to get to the specifics of the mission, etc.

As a conclusion, in this section we have seen in the explanation of ISIS, this UAS simulation platform provides an environment in which designed services can interact with others being designed. It provides an easier and safer way to test the mission application. ISIS minimizes both the costs of test development and validation, as well as providing easy migration of the software from the tested platform to the real flight platform.

My learning phase to familiarize myself with the ISIS environment consisted on programing very simple services (software components) that helped me to understand the subscription / publication concept. These services are in a “MAREA user manual” document developed by C. Barrado and J. Lopez [5], and it consists in create two software components that publish and subscribes data on MAREA to exchange data between them. Thanks to this manual I learned how to program services in the ISIS, understand how data flow in ISIS and prepared me for beginning with the eDEP integration on it.

1.1.2. earlyDemonstration & Evaluation Platform (eDEP)

In this section we will see a short summary of eDEP [6] according to the needs of our project. We will see the most relevant features of this simulation platform we used for integration with ISIS.

eDEP is an ATM simulation platform programmed in Java developed by software company Graffica under some requirements of Eurocontrol. eDEP focuses on simulating not only the positions of the air traffic controller but also simulates pilot positions (although our project will not be used).

The most relevant features are:

- ICARUS has the privileges of modify eDEP, allowing the platform to accommodate our needs. In our case, this fact has allowed us to know how to manipulate, extract and place air traffic.
- ATM simulator uses of event-driven simulation. This is a method that stands out because the evolution of the simulation results based on events that occur over time. eDEP uses the observer pattern, in which a change in one object makes its observers to notify the rest of the items concerned.

- This platform can generate Automatic Dependent Surveillance Broadcast messages (ADS-B messages). This feature will define how our project evolves. In the next section we explore in detail called Concepts of dealing with this type of message

The user can set the stage that he wants to create any simulation. Below we describe how to configure a simulation scenario. To create a scenario, three files are required: the airspace.dat, traffic.dat and simulation.gsdk. They are text files and they must follow a syntax established to define each of the components outlined below:

- Setting the airspace file, as its name suggests, is where you define all the attributes that adjust the setting to what we want, ie, all static elements of the simulation. It is a text file that must be saved with the extension ". dat."
- Flight plans: traffic.dat is also a text file with the extension ".dat" which it only includes the flight plans of aircraft that we want in the simulation.
- Simulation.gsdk indicate which services are being to run and in which configuration. Moreover, it sets the map file location, airspace, performances, and traffic.

In chapter 6, we will create two scenarios to perform an ISIS and eDEP integrated simulation. To create both simulations we set up a scenario with eDEP air traffic to introduce in ISIS. This chapter will show step by step how to create the files discussed in the previous paragraphs.

1.2. Concepts

As mentioned at the beginning of the chapter in this section we will see, in this order the concepts as Unmanned Aircraft Systems (UAS) [4], Automatic Dependent Surveillance Broadcast messages (ADSB messages) and ASTERIX standard. These concepts are needed property the project.

1.2.1. Unmanned Aircraft Systems (UAS)

An aerial vehicle flying without a pilot on board and without a recreational aim is commonly referred to as an Unmanned Aerial Vehicle (UAV). An UAV is an acronym that identifies an aircraft that can fly without a pilot; that is, an airframe and a computer system that combine sensors, GPS, servos and CPUs. All these elements have to pilot the plane with no human intervention.

However, this term is becoming substituted by Unmanned Aircraft System (UAS) [3]. UAS is being adopted by the highest ranking international organizations, like the International Civil Aviation Organization (ICAO), the European Aviation Safety Agency (EASA), EUROCONTROL and the U.S. Federal Aviation Administration (FAA), as the correct and official term. The changes in the acronym are due to the following aspects:

- The 'Unmanned' of UAS: refers to the fact that there is no pilot on the flying part of the system.
- The 'Aircraft' of UAS: this term was changed because the aviation authorities are only responsible for aircrafts and not in any way for flying aerial vehicles.
- The 'System' of UAS: this was introduced because UAS not just a vehicle, it is a system that includes other parts.

As we can see in the Figure 1.4, the UAS can be defined generically as a system divided into an air segment, a ground segment and Communications segment

- The segment air is formed by the aerial platform; the appropriate payload assigned to the mission and part of the system communication.
- The ground segment includes the system control aircraft and its payload, communications and station. This segment allows to disseminate information obtained sensors to different users either directly or through different networks of Command, Control, Communications and Computers.
- Communications Segment: this is divided into the Command & Control data link, the Payload data link and External Communications.

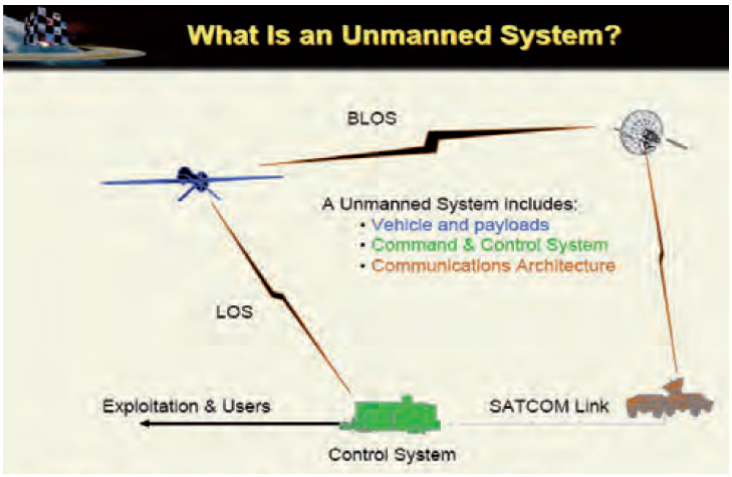

Figure 1.4 UAS segments

1.2.2. Automatic Dependent Surveillance Broadcast (ADS-B)

As the project progressed, we noticed that eDEP generated a type of messages that contained telemetry information. These messages are called ADS-B messages and are used in commercial aviation. For this reason these kinds of messages are important for the project.

The system of Automatic Dependent Surveillance (ADS) [7] is a set of tools and procedures for cooperative monitoring of air traffic control. The aircraft determines its position using a satellite positioning system (GNSS) and other positioning system. Through the ADS-B periodically sends this position to other nearby aircraft and ground stations. Unlike conventional surveillance systems, in which the aircraft position is determined directly from the ground station, with the ADS position measurements are performed on board, with navigation information, and then sent to monitoring centers. In the Figure 1.5 [8], we can see how ADS works.

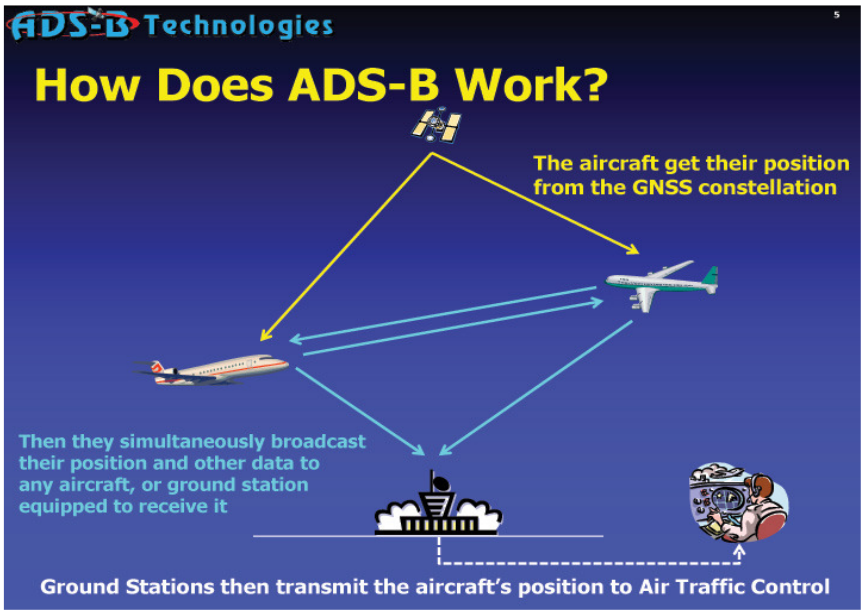

Figure 1.5. Automatic Dependent Surveillance

As we have seen, the ADS-B is a system of data exchange in the field of Surveillance. But, what format is? To overcome the differences between formats and protocols of each country in the late of eighties Eurocontrol defined a standard called ASTERIX (All Purpose STructured Eurocontrol Radar Information Exchange).

1.2.3. All Purpose STructured Eurocontrol Radar Information EXchange Messages (ASTERIX Messages)

In this section we present All Purpose STructured Eurocontrol Radar Information EXchange messages (ASTERIX messages). There is a very important concept because is the standard format of messages that exchange ISIS with eDEP.

The ASTERIX [9] messages are a Eurocontrol standard. It describes the message structure of the ADS-B. The standard was devised by the Study

Group on the exchange of surveillance related data between processors of ATC systems.

The ASTERIX standard was defined in the late eighties with the aim of overcoming the deficiencies among a variety of formats there is national and specific protocols and formats of specific vendors. Serving as an intermediary format, can carry all the current information in any format existing surveillance.

The purpose of this standard is to establish a homogeneous infrastructure European Air Traffic Control.

One of the central concepts of ASTERIX is the category of data. The format sets up to 255 categories each supplying the transfer of a specific class of related surveillance data.

Remember eDEP has the ability to generate messages ADS-B. For this reason, we focus on CAT021 [10] and CAT244 categories which are referred to ADS-B data format.

CHAPTER2. ARCHITECTURE

ISIS is an UAS simulation platform with many possibilities when designing missions. As we saw in the chapter "Previous work", this platform allows to program flight plans and implement them through an autopilot. We also explained that through designed software components (services) for particular tasks we could provide the functionality required UAS on a mission. When ICARUS group took eDEP, quickly they thought of a new functionality in ISIS. This new functionality should extract the air traffic generated in eDEP simulations, process and include them in ISIS to perform simulations of an UAS with air traffic.

This chapter aims to explain the architecture of the integration of eDEP in ISIS. That is, we explain the components that perform some tasks of computer interfaces and communication between them.

At the beginning of the project, two services were implemented that were in charge of establishing the communications with eDEP. The messages used in the data exchange were telemetry information (Position, speed and bearing). But we found that doing this was inefficient and decided to change the course of the project introducing a single service that receive and sent data.

Moreover, eDEP generates ADS-B messages. ISIS did not use ADS-B messages because we had not enough knowledge about them. This kind of messages has the advantage that they are able to transmit bi-directionally. In the Objectives section, one of the targets was to homogenize data, with ADS-B messages, we achieve this purpose. For this reason, we implanted the second evolution on ISIS. It consist in extract the eDEP air traffic using ADS-B messages.

But ISIS is a project in constant evolution. We realized that if we could extract air traffic simulation platform as eDEP, it could also be extracted from other simulators. Therefore, there is a third evolution of ISIS is under construction. The third evolution is not part of this project, but will show the amount of possibilities opened up for the future.

So, we see that this project is constantly evolving. For this reason, there are three evolutions of the project which we will explain during this chapter.

2.1. First evolution of the integration architecture

The mission of this evolution was in one hand extract the eDEP telemetry and send it to ISIS through an our designed software component called “Telemetry Publisher From Edep”. On the other hand we create other service called “Telemetry Monitor To Edep” that sends the UAS telemetry from ISIS to eDEP.

The Figure 2.1 shows the general architecture of the first evolution of the integration:

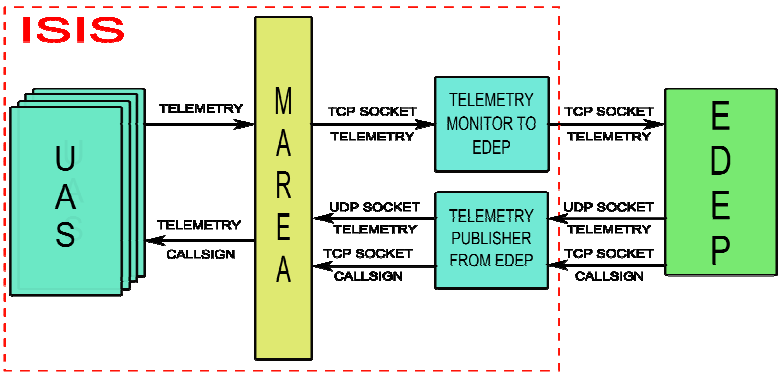

Figure 2.1. Architecture of the first evolution

Once introduced the two software components that we designed early in the project, then we explain in more detail the “Telemetry Monitor To Edep” and “Telemetry Publisher From Edep”.

Telemetry Monitor To eDEP

This software component (service) called “Telemetry Monitor To eDEP” is responsible of transmitting telemetry data from UAS simulated in ISIS to eDEP. The telemetry data are composed by latitude, longitude, altitude, speed and bearing.

At first we thought it was vitally important that the telemetry of the UAS simulated on ISIS arrived to eDEP correctly. So we needed that the transmissions of telemetry data will be held through a connection that ensured the reception in eDEP of all telemetry. That is why we used a connection type called TCP socket. This kind of connection ensures the complete reception of data sent through a link that asks if the data has reached into every transmission.

Recall that the services (software components) in ISIS access to MAREA through the philosophy Publication/Subscription. The “Telemetry Monitor To Edep” service demands (subscribes) to the telemetry data of the simulated UAV through TCP socket connected to MAREA. The telemetry is processed on “Telemetry Monitor To Edep Service” and is converted into bytes to be sent to eDEP through another TCP socket. eDEP should include the UAV in the air traffic as an aircraft more.

Telemetry Publisher from eDEP Service

“Telemetry Publisher from eDEP” extracts the air traffic from eDEP and sends it to the UAS simulation in ISIS. The format of the telemetry message is position (latitude, longitude, and altitude), speed, bearing and callsign (aircraft register). The callsign is needed to identify the aircrafts from eDEP interacting with our UAS simulated in ISIS.

This service gets the telemetry (position, bearing and speed) through a UDP socket and callsign through TCP socket. The frequency of the reception of telemetry data is high, for this reason the transmission is made through an UDP socket even if it does not ensure the reception of data, but is more efficient than TCP. On the other hand, the transmission of the callsign is essential and it is necessary ensure the reception. For this reason it works through TCP socket.

The “Telemetry Publisher from eDEP” is who demands data (telemetry and callsign) acts as a publisher on MAREA because is who has data.

2.2 Second evolution of the integration architecture

The second evolution of the Integration received the name of “Traffic Service”. This evolution was designed when we discovered that eDEP was capable of generating ADS-B messages. Furthermore, this type of messages enables bidirectional communication. For this reason, the second evolution of the integration is focused on the use of ADS-B messages because it is a standard protocol.

Now the “Traffic Service” is a single service, not two as the first version. In this service the data flows in two directions: eDEP to ISIS and ISIS to eDEP. In the 2.2, we can see the architecture of the second version of the project

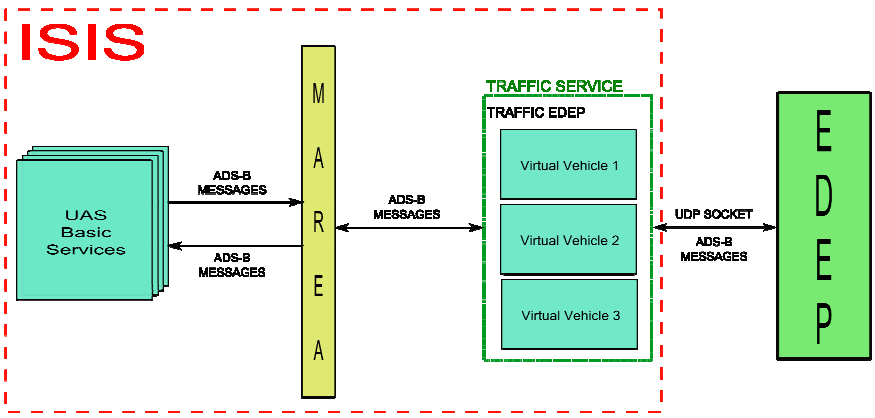

Figure 2.2. Architecture of the second evolution

Data is transmitted through an UDP socket because the frequency of the acquisition from eDEP is high, but it is not necessary ensure the connection. By connecting both platforms with a UDP socket, we gain speed transmission of telemetry data. In the case of losing telemetry data packet, we do not care because the transmission rate is so high that immediately come another.
The “Traffic Service” extracts the air traffic from eDEP to include them in the UAS simulation in a Virtual Vehicles form. For example, if a user is simulating an UAS with Flight Gear and he activates the “Traffic Service”, he could receive telemetry data from the aircrafts generated from eDEP's air traffic and other services can manage this data. As has been said before, the ADS-B messages allow a bidirectional communication, this is, and eDEP receives this kind of messages, processes them and includes the UAS in its air traffic.

This evolution of ISIS meets the objectives set at the beginning of the project.

2.3 Third version Second evolution of the integration architecture

This ISIS improvement is not part of my TFC. But it shows that the second evolution, and the one presented this project, is designed to leave open the possibility for further developments without having to replace anything.

This is the last evolution of the “Traffic Service” and it is in development. The key idea of this version is to have multiple simulators simultaneously generating air traffic and sending it to ISIS. Recall that in the second improvement eDEP generated air traffic to interact with our UAS, the third evolution of the “Traffic Service” follows the same idea but now are different simulators (eDEP, X-PLANE, Flight Gear, etc.) generating traffic at the same time.

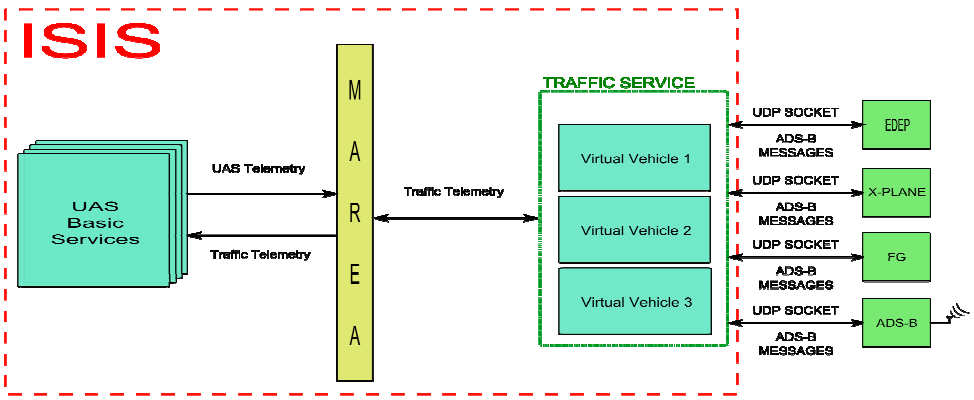

Figure 2.3. Architecture of the third evolution

The figure 2.3 shows the architecture of the third improvement. All the transmissions are through UDP sockets using ADS-B messages. The “Traffic

Service" processes the ADS-B messages of the air traffic from the simulators and the ADS-B antenna to interact with the UAS allowing a good simulation.

Moreover, there is a new functionality related with ADS-B messages. In this version, ISIS is able to acquire real air traffic through ADS-B antenna and include it in the UAS simulation.

CHAPTER3. DESIGN

In the previous chapter we saw the arrangement of software components of each evolution and how it is the communication between ISIS and eDEP. This chapter describes in detail how services are designed, i.e. the processing of messages or data within the software components that I designed.

3.1 First evolution of the integration design

During the first weeks from the start of the integration of this project we created two services on the UAS simulation platform. We wanted to exchange messages with eDEP telemetry. So, we designed the structure of messages and the way exchange between the two platforms. In this section we explain the design of “Telemetry Monitor to eDEP” and “Telemetry Publisher from eDEP” services.

Telemetry Monitor to eDEP Service

eDEP needs detect the UAS simulated in ISIS. In order to be property integrated into ISIS, “Telemetry Monitor to eDEP” acts as a Subscriber (demands the UAS data) in order to send it to eDEP. The figure 3.1 illustrates the way to order the callsign and the telemetry.

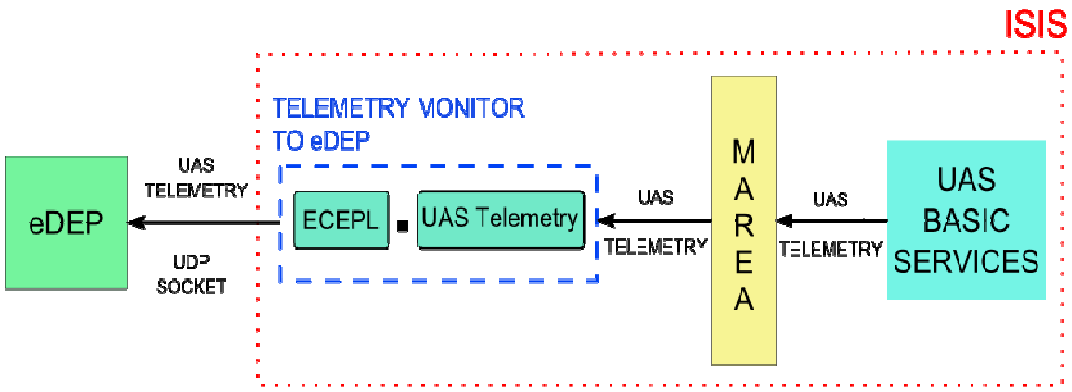

Figure 3.1. Design of the first evolution of the integration

The message consists in the registration of the simulated UAS (in our case ECEPL) followed by a dot (“.”) and the telemetry (latitude, longitude, altitude, speed and bearing). The figure 3.1 illustrates the way to order the callsign and the telemetry.

Telemetry Publisher from eDEP Service

“Telemetry Publisher from eDEP” service takes the callsing and the telemetry from eDEP through TCP and UDP socket respectively. UAS services demand

the data (subscriber) and “Telemetry Publisher from eDEP” provides the information (publisher). So the service extracts the callsign of each aircraft from eDEP and all their associated telemetry data. Thus, the callsign acts as a header of the message followed by the telemetry (position, speed and bearing). In the Figure 3.2, it shows a scheme of design of the “Telemetry Publisher from eDEP” service:

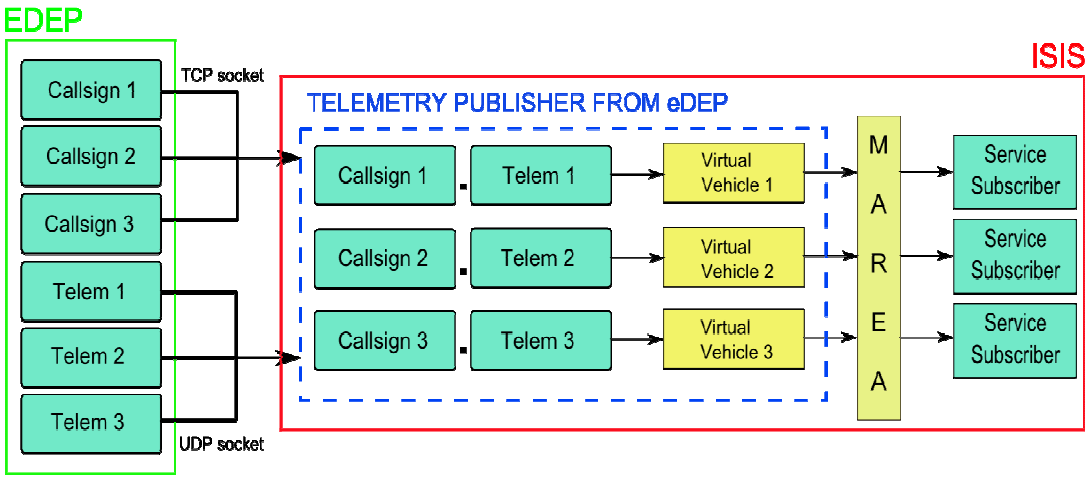

Figure 3.2. Design of the Telemetry Publisher from EDEP Service

The “Telemetry Publisher from eDEP” assigns a callsing received from eDEP to an aircraft (virtual vehicle) and all their associates telemetry, processes and groups orderly the data (callsign and their telemetry) and publishes them for all subscribers services.

3.2 Second evolution of the integration design

The second improvement of ISIS consists in an implementation of the “Traffic Service”. This software component is focused on ADS-B messages because eDEP generates them. For this reason, it has changed the architecture and the design respect the first evolution. In this way, ISIS through the “Traffic Service” can process ADS-B messages.

Through an UDP Socket, “Traffic Service” extracts the ADS-B messages from eDEP. But it's necessary a transcoder to traduce these messages in order to ISIS can understand them because ISIS process telemetry data. Then ADS-B messages goes to a Transcoder to convert this kind of messages (following the standard format ASTERIX CAT021 and CAT244) to telemetry and Callsign messages. The scheme of the design of the second version is represented in the Figure 3.3:

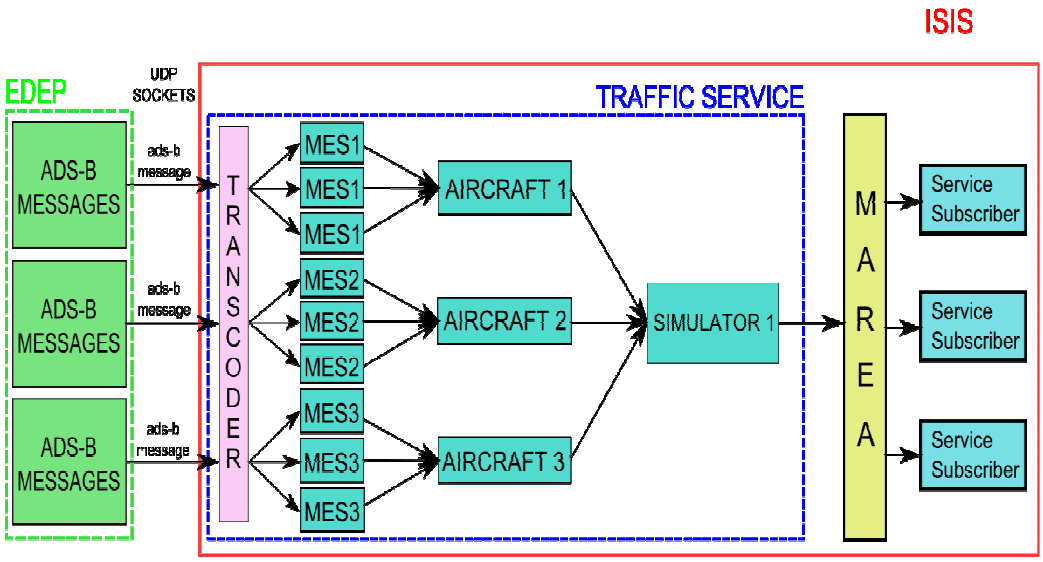

Figure 3.3. Scheme of the design of the second improvement

Once the transcoder has translated the ADS-B messages in Telemetry and Callsign data, the "Traffic service" is responsible for group these traduced messages (MESS in the Figure 3.2) and generates an aircraft with them. That is, a group of messages (now Telemetry and Callsign messages) generate one aircraft.

Now, to maintain the order and have centralized data, all the aircrafts generated conform a Simulator (in this version of the project is eDEP Simulator) which it will publish in MAREA the air traffic from eDEP (aircrafts generated by messages) to the all subscriber services that demands the data through MAREA.

3.3 Third evolution of the integration design

The third evolution is in implementation phase. The "Traffic Service" design is the same as that of the second but with a difference: The air traffic is taken by ISIS from various simulators.

The idea is activate various simulators at the same time or separately and extract their air traffic to make a good UAS simulation. These simulators are: eDEP, X-PLANE [2](Flight Simulator), Flight Gear and real air traffic taken with an ADS-B antenna. The scheme of the design of the third version is represented in the Figure 3.4:

[2] X-Plane is a flight simulator produced by Laminar Research. X-Plane is packaged with other software to build and customize aircraft and scenery, offering a complete flight simulation environment. X-Plane also has a plugin architecture that allows users to create their own modules, extending the functionality of the software by letting users create their own worlds or replicas of places on earth.

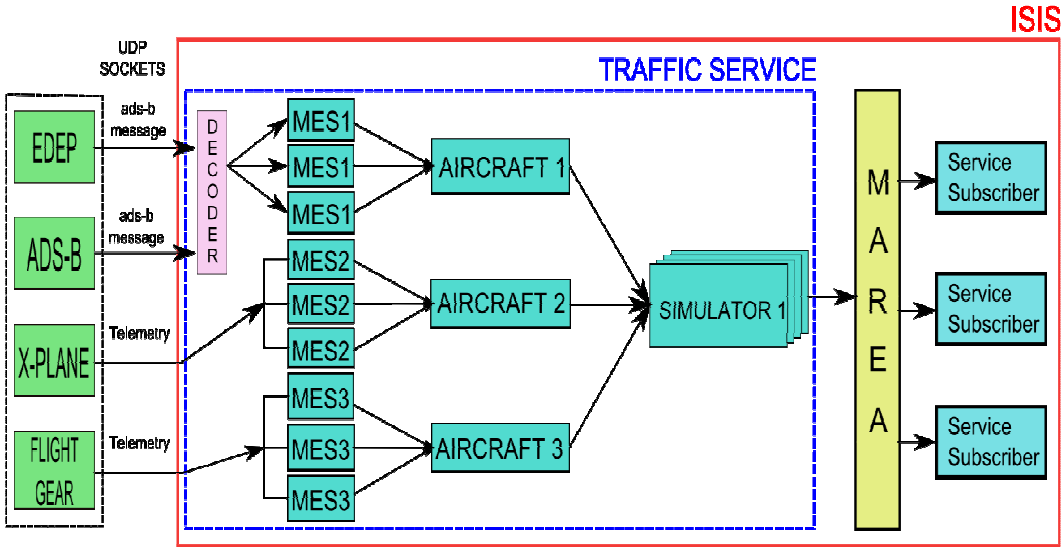

Figure 3.4. Scheme of the design of the third evolution of ISIS

eDEP and ADS-B antenna generate ADS-B messages whereas Flight Gear and XPLANE generate telemetry messages (position, speed, etc.) which ISIS can understand without Decoder.

From eDEP and ADS-B antenna, the “Traffic Service” extracts air traffic ADS-B messages through UDP socket to a Decoder located in the “Traffic Service”. This software component groups the traduced messages (MESS in the Figure 3.4) of each simulator in an orderly manner avoiding mixing messages between simulators. That is for example, all the messages from eDEP will go together. Therefore, if the “Traffic Service” groups the eDEP messages will generate eDEP aircrafts.

A set of aircraft of a particular simulator will generate a particular Simulator in ISIS. For example, if we have activated eDEP and X-Plane at the same time and we want extract the air traffic from both to interact with the simulated UAS; Traffic Service will create two simulators: one for eDEP and other for X-PLANE.

The “Traffic Service” is structured as the figure 3.5 shows. There are three levels: ISIMULATOR (Higher level), IEXTERNALAIRCRAFT (Medium level) and IMESSAGE (lower level).

The “Traffic Service” implements inherited and abstract classes, thus we avoid repeat code in all blocks because this kind of classes inherit all the methods and allows add new of them.

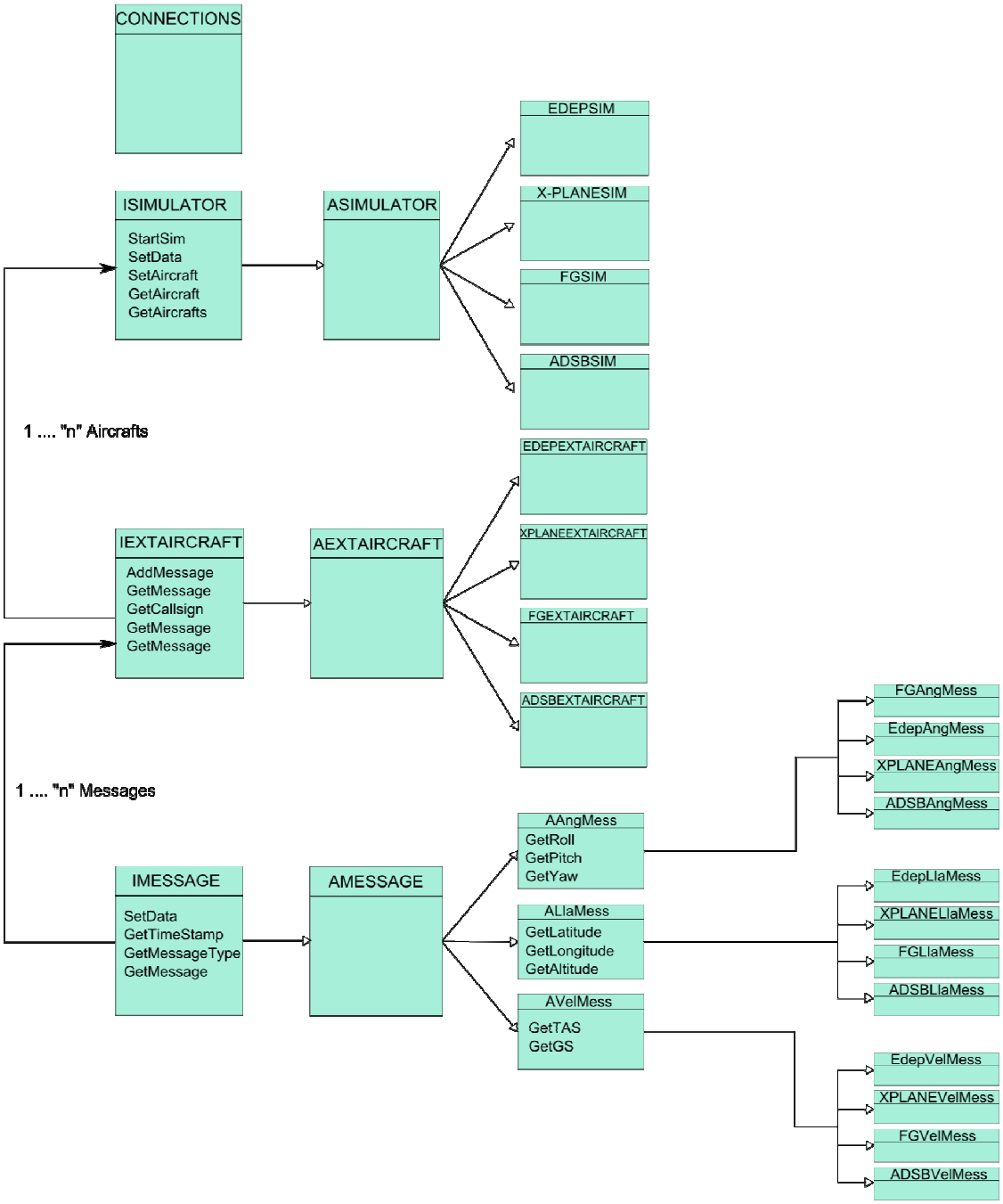

Figure 3.5. Class Diagram of the Traffic Service

A group of Messages (from IMESSAGE) forms an Aircraft, and a group of Aircrafts (IEXTERNALAIRCRAFT) forms a Simulator.

AMessage is an Abstract Class (inherits all methods of IMessage) that groups the data from the AAngMess, ALlamess and AVelmess (inherited classes from AMessage) and in turn, there are inherited classes from this classes for each simulator (EDEPAngmess, XPLANEAngmess, etc.). This program structure allows grouping the messages from each simulator, for example, if we are working with eDEP (see the figure 3.5), data flow at follows: EdepAngMess,

EdepVelmess and EdepLlamess takes from eDEP the data (angles, Speed and Position respectively), and AMESSAGE group and give them to the EdepExtlAircraft (inherited class from AEXTAircraft). The Simulator class is responsible of group each aircraft in their Simulator. In this manner, Traffic Service can work with different platforms of simulation at the same time.

There is another class called Connections which is responsible for making the Ethernet connections between the Traffic Simulators and ISIS through UDP Sockets.

CHAPTER4. IMPLEMENTATION

The programming of this project has not been performed once. It has been necessary to program small parts to check they worked properly. Once verified each party to their tasks properly, we proceeded to their assembly with other parts.

During this chapter we will explain how has been the process of the programming of the first and the second evolution of ISIS. The third improvement is still in the programming phase. For this reason, below we will describe in the following sections how it has made the implementation of the project eDEP integration of in ISIS.

4.1 First evolution of the integration implementation

As we have seen in previous chapters, the first evolution had 2 services: “Telemetry Monitor To Edep” and “Telemetry Publisher from eDEP”.

Before working directly on eDEP, it was necessary to know if the transmission of data was correctly in a “test file” called FakeEDEP. FakeEDEP had two functions (“Telemetry Monitor to eDEP” and “Telemetry Publisher from eDEP”) as we can see in the Figure 4.1:

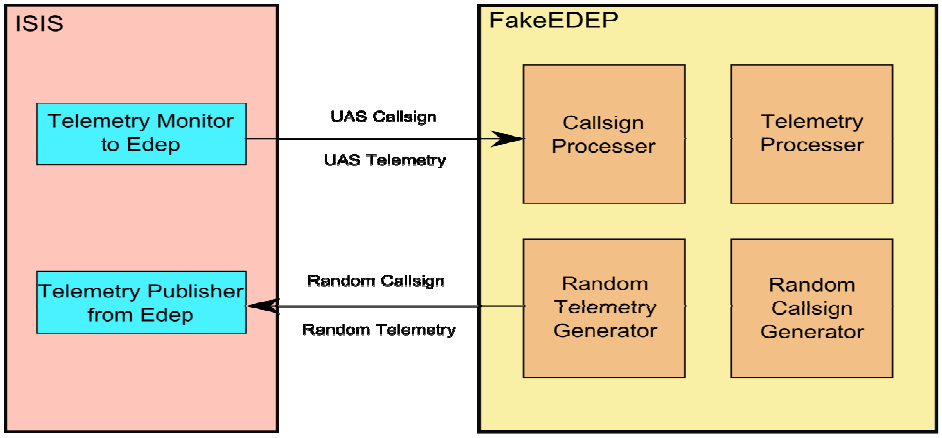

Figure 4.1. Implementation of the first evolution

- On the one hand receive and process the simulated UAS data (Callsign and Telemetry) from ISIS correctly. Thus, we would know if the programed sockets worked properly, then it would be easy modify the code to connect ISIS with eDEP.

- On the other hand, FakeEDEP had to be able to generate random Callsign and Telemetry to test if the sockets worked correctly and the data arrives to ISIS.

All tests on FakeEDEP were correct, FakeEDEP received the telemetry of the UAS (using Flight Gear Simulator) from ISIS correctly and the Random Generator data transfer (Callsign and Telemetry) worked. This part of the project lasted approximately 1 month.

But the knowledge learned about ADS-B messages changed the architecture and the design of the project, giving way to the second version. Moreover, it wasn`t necessary two services for the program (one for receive and other to send data) and the code was disordered. For this reason the project was rebuilt.

4.2 Second evolution of the integration implementation

The way of implementation of the second evolution is different than the first version. In this case the program has been implemented in parts:

- Structure of the Traffic Service
- Process of ADSB Messages
- Group all parts

First, it was implementing the structure of the “Traffic Service” in a project called “FakeTrafficService” (Figure 4.2), that is to say how process the data (the intelligence of the Service). Moreover, prepare the Connections Block (UDP sockets) for all data transmissions between ISIS and eDEP. Later we prepare the code to link to ISIS.

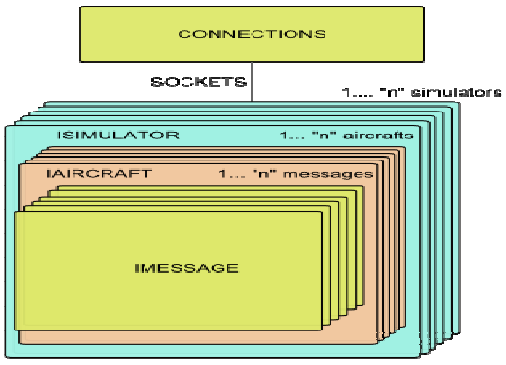

Figure 4.2. FakeTrafficService scheme

Process ADSB messages was the next step, so it was programed a project called “AsterixCat244Test” as shows the Figure 4.3 where it was implemented.

the CAT021 and CAT244 messages processing. The implementation consists of processing the ADSB messages and converting in telemetry using a decoder. This allowed to ISIS could understand this kind of messages. In the following figure, it shows all methods that allows process the ADSB messages.

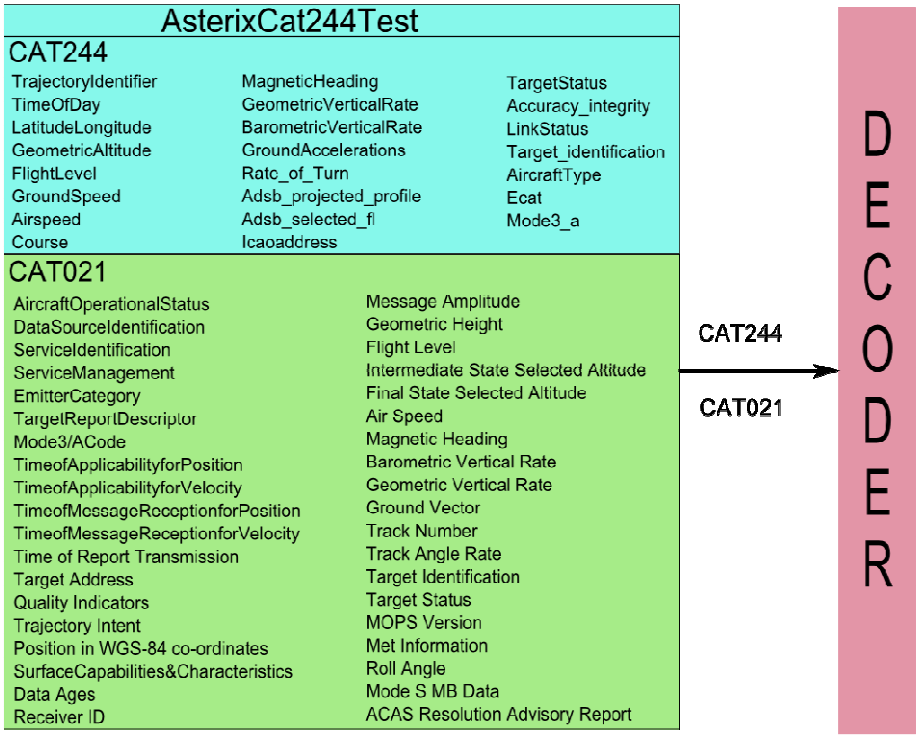

Figure 4.3. AsterixCat244Test scheme

This part of the implementation gave me knowledge of bit-level programming. This is so because eDEP is programmed in two different programming languages and it was necessary work with bits.

To finish the implementation, it was necessary group all parts (FakeTrafficService and AsterixCat244Test) and adapt to ISIS in a service called “eDEPTrafficManager”. This Service implements the actions for the communication between ISIS and eDEP.

CHAPTER5. SIMULATIONS

This chapter describes the applications of the “Traffic Service” through two cases of simulations: VFR (Visual Flight Rules) and IFR (Instrumental Flight Rules). It will explain, step by step, how to prepare the simulation. The simulations discussed in this chapter will take place at Sabadell Airport.

5.1 Case 1: UAS Simulation in VFR procedures

To make the simulation with VFR procedures, first we must create the scenario with the necessary files in eDEP and second prepare ISIS to the test. Recall that ISIS uses Flight Gear (flight simulator) to display the simulation. Next, we explain how to create the eDEP files to the simulation and the actions to be taken in ISIS.

5.1.1. Creating eDEP scenario to the Simulation

Recall that eDEP is programmed in Java, for this reason it’s necessary open ECLIPSE (Java compiler) and select a correct Workspace (our case is eDEP) .To begin we must create three new archives: the “airspace.dat”, the “airtraffic.dat” and an “archive.gsdk” that contains the simulation engine.

Setting the “airspace.dat”:

The “airspace.dat” contains a region of the world where the simulations will be developed and the points where the aircrafts will pass. The first step is to select the region where the simulation will do, in our case it will be in Sabadell airport. Using a GoogleEarth application we select the region taking the latitude and longitude coordinates. It is necessary convert these coordinates from Degrees, minutes and seconds to degrees, because eDEP process degrees only. In the VFR simulation we chose for Airport Sabadell Region eight points that correspond to the cardinal points (N, NW, W, SW, S….etc.). From each point selected will appear an aircraft. The Figure 5.1 illustrates the points taken around the Sabadell Airport.

Figure 5.1. Points around of Airport Sabadell

VFR procedures require additional waypoints that allows landing. These points are Crosswind, Downwind and Base as the Figure 5.2 shows:

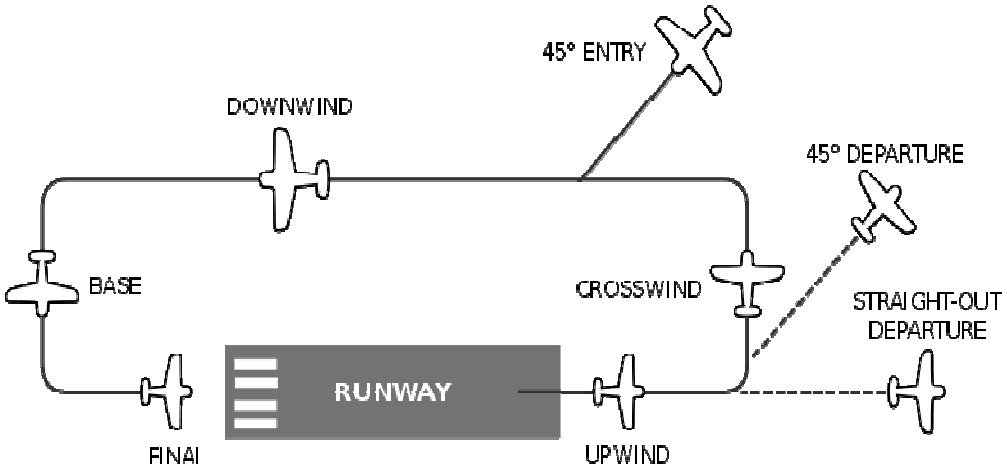

Figure 5.2. VFR procedures

In the next step, we put the VFR procedures points in GoogleEarth application to know their coordinates (latitude, longitude) for converting the minutes and seconds to degrees. The Figure 5.3 illustrates the VFR procedures in the Airport of Sabadell:

Figure 5.3. VFR procedures in the Airport of Sabadell

Once done, it only has to configure the file "airspace.dat". Following this route in ECLIPSE: "eDEP/ATC/src/atcapp/resources/demos/ians" we access to demos of eDEP. Here, we create the "airspace.dat" (Charlyairspace in our case) clicking with the right button of the mouse and selecting add new file.

The first part of the file is data about of the region, we write all the coordinates that we extract from GoogleEarth (N, NW, W, SW, S, SE, E and NE). The second part is information about our waypoints (Region points described before and the coordinates of Crosswind, Downwind and Base). The third part is the information of the Sabadell airport (his coordinates and all frequencies like ILS). The last part of the file is longitude and latitude data about the origin airport of the aircrafts that participate in the simulation. The Figure 5.4 illustrates the parts described.

REGION/SECTOR

```
SECTOR PRUEBA
   REGION
   ALTITUDE 1000 5000
   41.578914    2.055614   //INITIALN
   41.546069    1.570653   //INITIALNW
   41.496325    1.949750   //INITIALW
   41.417725    2.634541   //INITIALSW
   41.427239    2.217575   //INITIALS
   41.502203    2.282600   //INITIALSE
   41.574017    2.331072   //INITIALE
   41.619756    2.254744   //INITIALNE
END ATC
```

AIRPORT

```
RUNWAY LELL_13 41.523344   2.10073056 900 128
```

ORIGIN AIRPORTS

```
//AEROPUERTO SABADELL

AIRPORT LELL  41.520833   2.1050
COMPRISING
END

//AEROPUERTOS ORIGEN

AIRPORT EDDV  52.4603   9.6836
COMPRISING
END

AIRPORT EDDT  52.5611  13.2894
COMPRISING
END

AIRPORT EDDL  51.2808   6.7572
COMPRISING

END

AIRPORT EDDF  50.0341   8.5714
COMPRISING
END
```

ALL POINTS

```
FIX INITIALN     41.578914  2.055614
FIX INITIALNW    41.546069  1.570653
FIX INITIALW     41.496325  1.949750
FIX INITIALSW    41.417725  2.634541
FIX INITIALS     41.427239  2.217575
FIX INITIALSE    41.502203  2.282600
FIX INITIALE     41.574017  2.331072
FIX INITIALNE    41.619756  2.254744
FIX CROSSWIND    41.511864  2.122022
FIX DOWNWIND0    41.514939  2.133705
FIX DOWNWIND1    41.523794  2.122372
FIX DOWNWIND2    41.536589  2.106793
FIX BASE         41.529709  2.088956
FIX SISTA13      41.523344  2.10073056
```

Figure 5.4. Airspace.dat

Setting the "airtraffic.dat"

The airspace.dat file contains all information about the aircrafts that make up the air traffic of the simulation and their flight plans. In this test, we used sixteen aircrafts with their corresponding callsigns (CHARLY1, CHARLY2, CHARLY3….). This file allows set the model of the aircraft, the start time of the simulation of each aircraft, his callsign and speed, and form a Flight Plan using all points created in the "airspace.dat" file.

All aircraft models are in a file called "badatypes.dat" and his access route in Eclipse is "eDEP/ATC/src/atcapp/resources/common/atc/badatypes.dat". For this simulation we have chosen aircraft with ENGINE TYPE PISTON because the region and the airport are too small for an aircrafts with a engines type jet or turboprop (they are too faster to this kind of airport). The airtraffic.dat file is illustrated as follows in the Figure 5.5:

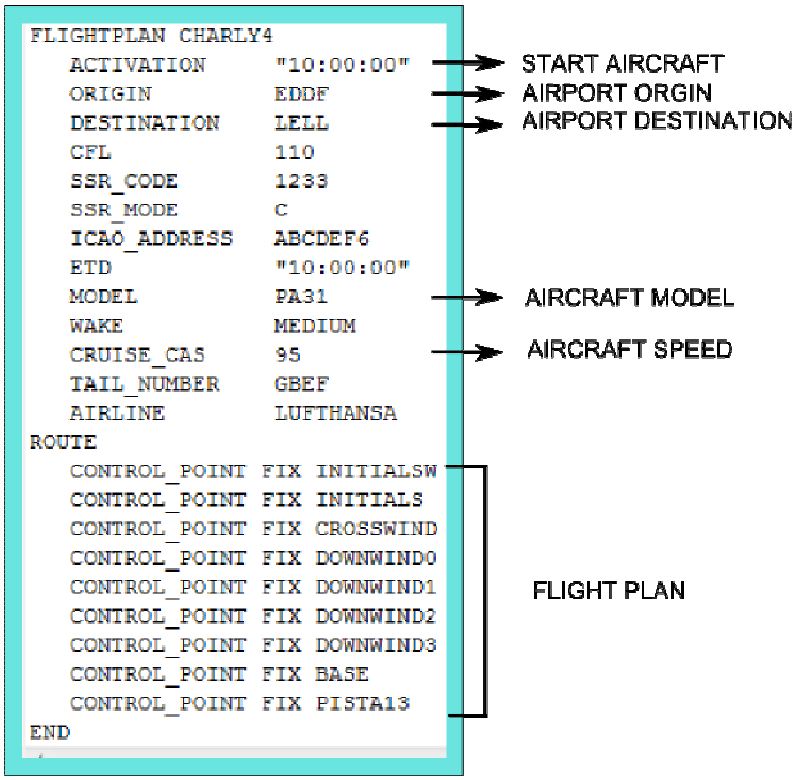

Figure 5.5. Example of airtraffic.dat

Setting the "demo.gsdk"

This file is the engine of the simulation; it is responsible of load the scenario through the "airspace.dat" and "airtraffic.dat" files. To set the test, we must copy the route of these files and modify the frequencies that will use (in our case the airport Sabadell frequencies) as follows in the Figure 5.6:

ROUTE OF THE FILES TO LOAD

```
ASP.SCENARIO          "atcapp/resources/demos/ians/charlyairspace.dat"
IFPL.SCENARIO         "atcapp/resources/demos/ians/charlytraffic.dat"
```

FREQUENCIES TO USE

```
PWP_RUHR.FREQUENCY ( 120.800 )
```

Figure 5.6. Set of demo.gsdk

Finally, we must start the simulation on eDEP using ECLIPSE to make the necessary corrections (avoiding collisions between aircrafts). For this purpose, it's necessary click on the debug button in ECLIPSE to start eDEP. With the time window, it's possible controlling the clock of the simulation. Increasing the time, you can accelerate the simulation. The Figure 5.7 shows an example of eDEP simulation:

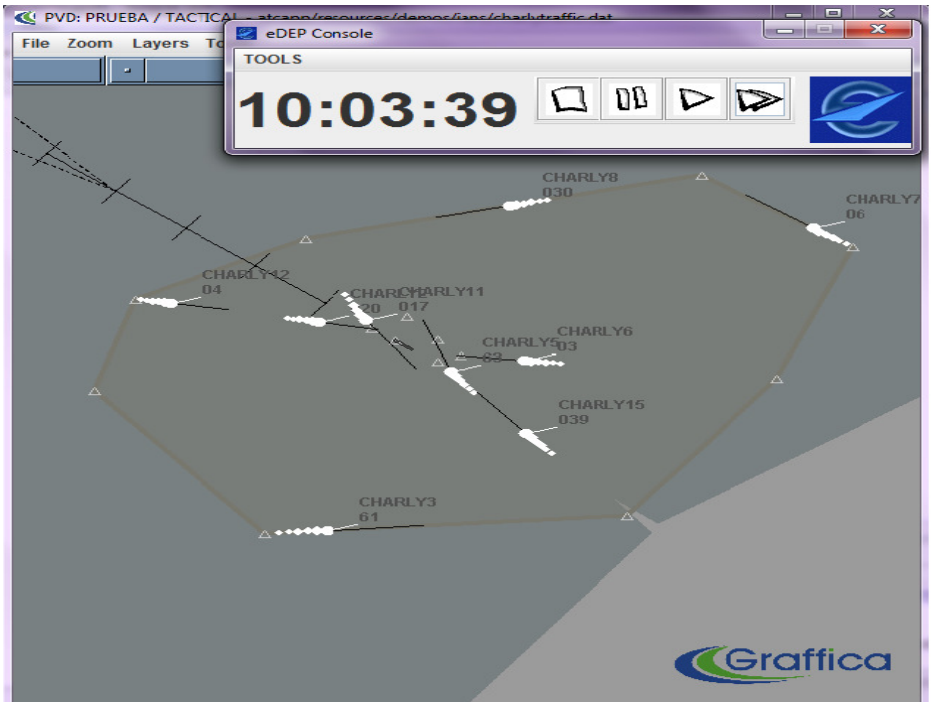

Figure 5.7. eDEP simulation in the airport of Sabadell

We have created a eDEP scenario. For our simulation of the eDEP of integration in ISIS, we have prepared a set of sixteen aircrafts doing approximation maneuvers to Sabadell Airport following the points of VFR rules (Crosswind, downwind and Base points).

To complete the simulation of integration, only need to prepare ISIS.

5.1.2. Preparing ISIS to the simulation

Before starting to explain how to prepare ISIS to the simulation, it is necessary to mention the Multiplayer application of FG (FlightGear). With the multiplayer function of FlightGear we are able to see other pilots and vice-versa. In our case, the other pilots will be the air traffic created by eDEP.

Architecture of the simulation system

To introduce air traffic from eDEP to ISIS and show it on screen through Flight Gear (FG), we need a MultiplayerFG that allows to create a networking Flight Gear session with several players thanks a server. The Figure 5.8 illustrates the architecture of the integration eDEP in ISIS:

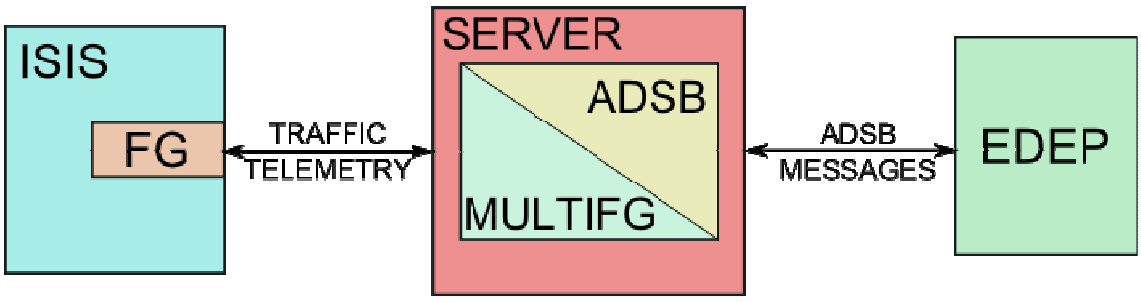

Figure 5.8. Architecture of the integration eDEP in ISIS

We have created a decoder in MultiplayerFG (Multiplayer Flight Gear) to translate ADSB messages in telemetry format because FG cannot process them. Each aircraft from eDEP is interpreted as a player thanks to MultiplayerFG. Traffic data from eDEP goes to ISIS through a server and thus our UAS can interact with the air traffic from eDEP.

Creating a Flight Plan

Now, we prepare a flight plan to our UAS. The key idea is that our aircraft flies following the VFR points (CROSSWIND, DOWNIND AND BASE) created in “airspace.dat” eDEP file. To do this action, we must create a flight plan file with a “Notepad” or “Wordpad” program putting “.xml” as extension. The figure 5.9 illustrates how set this file:

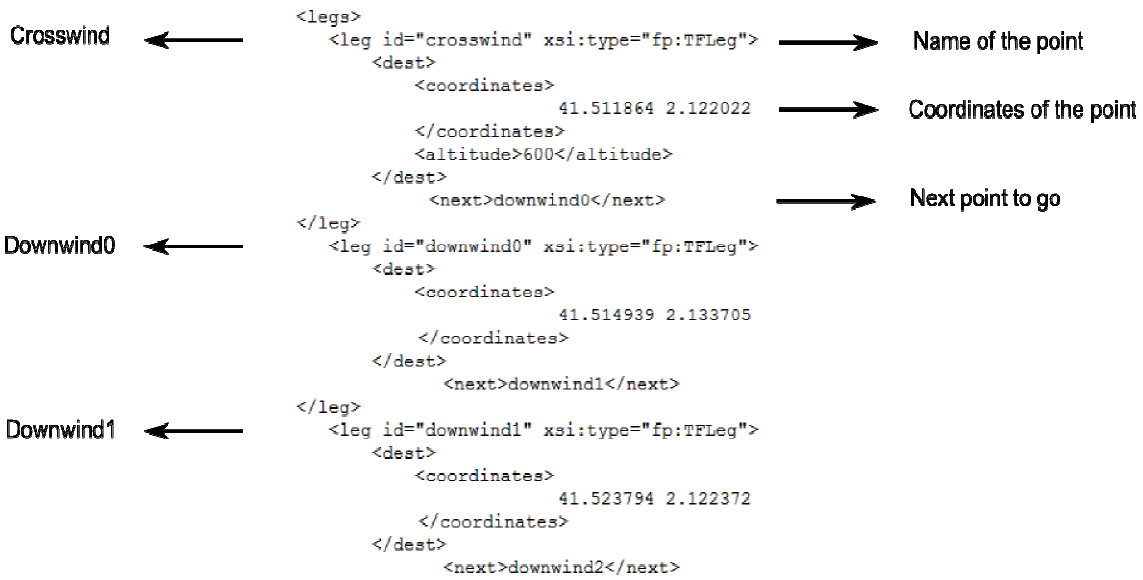

Figure 5.9. FlightPlan.xml

Starting the Simulation

Now is turn to start ISIS. The Figure 5.10 shows the console that appears when ISIS starts:

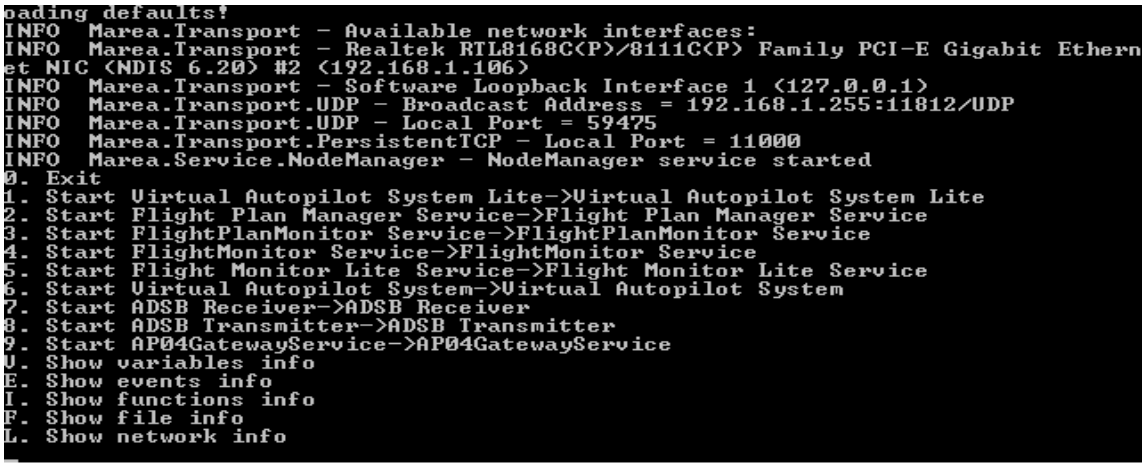

Figure 5.10. Console of ISIS

For our simulation, we activate the following options in the Console of ISIS:

- Press number 1:

  This is a service that provides a standardized interface to the particular autopilot on board. The Figure 5.11 illustrates this display and the red numbers the order in which options are enabled for our simulation:

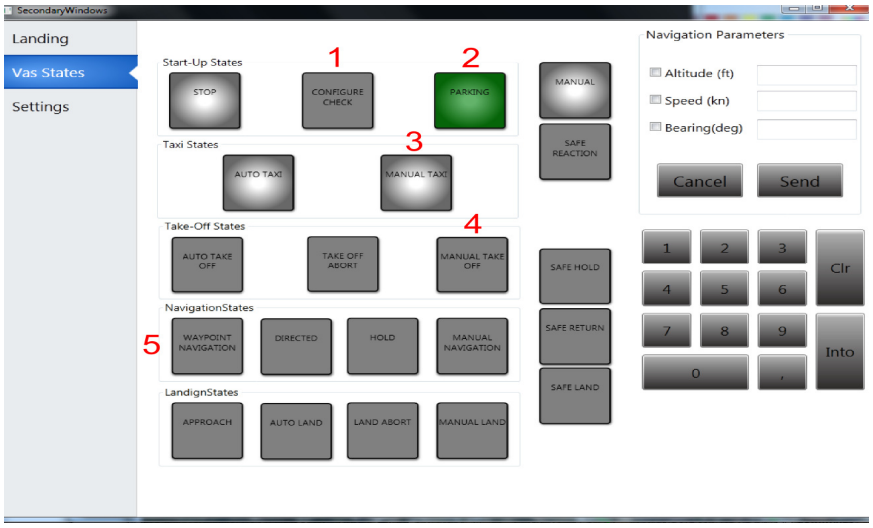

Figure 5.11. Autopilot Configuration

- Press number 3:

  This option allows loading the Flight Plan designed before. The Figure 5.12 shows the display of this option:

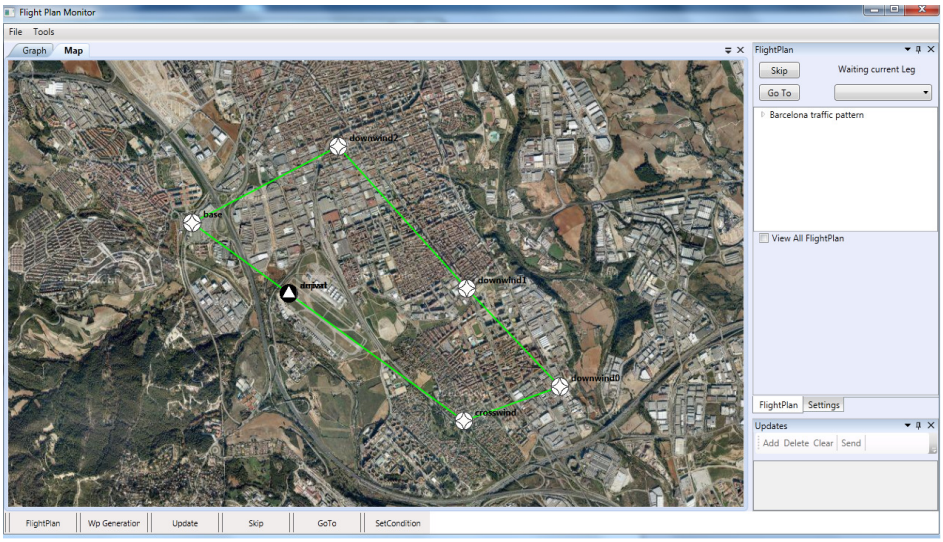

Figure 5.12. Flight Plan Monitor Service

- Press number 5 to start:

  This service is responsible for interacting with the pilot in command so that he or she can keep track of the mission in real time. That is to say, this service activates Flight Gear.

- Press number 7 y 8 to start ADSB receiver and transmitter respectively:

This option enables the ability to receive/transmit and understand the ADSB messages.

At this point you can start the simulation. Thanks to the scheduled flight plan, we can begin to perform all tests and check the previous studies with measurements of the simulation. The figure 5.13 and 5.14 show a captures of our VFR simulation:

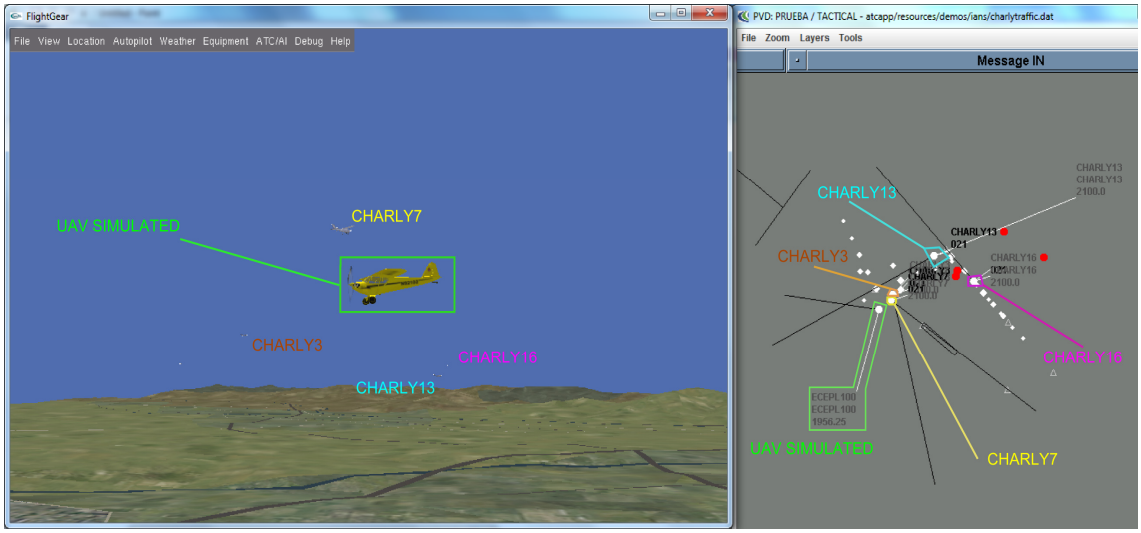

Figure 5.13. VFR simulation in eDEP integration in ISIS capture

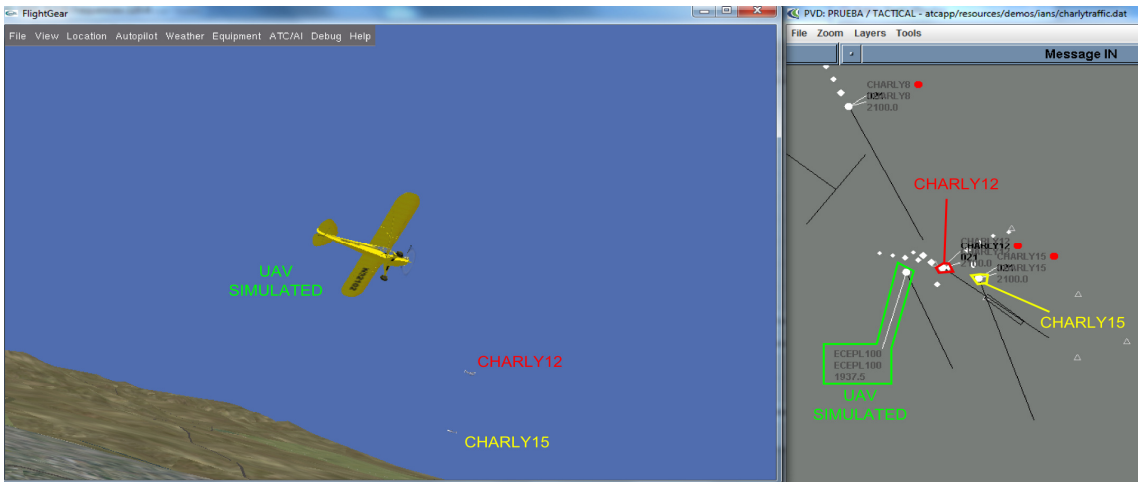

Figure 5.14. VFR simulation in eDEP integration in ISIS capture

At the right of the image 6.14, we can see eDEP simulation including our UAS in her air traffic and the left side of the figure appears Flight Gear with the air traffic from eDEP thus demonstrating that the integration between the two programs is possible, fulfilling the objectives of the project.

5.2 Case 1: UAS Simulation in IFR procedures

The IFR simulation is very similar to VFR. The mean difference is the absence MultiplayerFG. Now eDEP sends ADSB messages directly to ISIS. Each aircraft of the air traffic generated by eDEP is understood as a Virtual Vehicle (VV) in ISIS. Our UAV sends his position, speed, etc. in telemetry format to eDEP (not ASDB format) because eDEP can process this kind of data. In addition, ISIS executes eDEP directly (Figure 5.15).

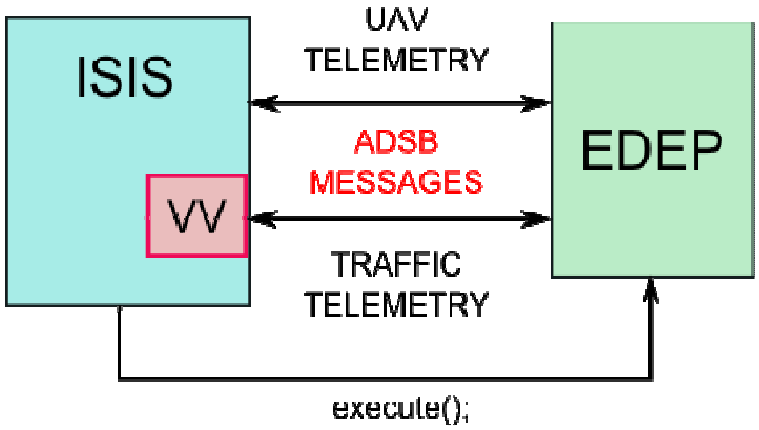

Figure 5.15. Architecture of IFR Simulation in eDEP integration in ISIS

The IFR simulation of the eDEP integration in ISIS has been designed in Düsseldorf Airport. To include our UAS in eDEP simulated scenario, from ISIS through Flight Gear (FG) we have load Düsseldorf Airport. One of options of FG is load any region of the world to perform a simulation.

In eDEP we have load a luxairspace.dat, luxtraffic.dat and luxdemo.gsdk following the “eDEP/ATC/src/atcapp/resources/demos/ians” route in eDEP. These files conform an eDEP scenario designed. Now we start the ISIS platform following the steps that we have seen in the section before.

As its name suggests, IFR refers to flight instrumental, for this reason in FG the user does not see the traffic generated by eDEP physically, but receives all their telemetry. If we chose other aircraft model with surveillance instruments, we can see the traffic through in their displays. We can see captures of the IFR simulation in Düsseldorf Airport. The right side of the figure 5.16 is eDEP display. In blue line, we can see our UAS simulated interacting with eDEP air traffic. At the right side is the UAS simulated in ISIS in the eDEP scenario.

Figure 5.16. IFR simulation in eDEP integration in ISIS capture

The VFR and IFR simulations in the eDEP integration in ISIS have accomplished the objectives of the project. Now we can simulate an UAS from ISIS platform interacting with air traffic generated by eDEP. Now the ICARUS group can perform more complex operations in which the UAS interact not only the environment but also with air traffic.

CHAPTER6. CONCLUSIONS AND PROPOSALS FOR IMPROVEMENT

During the explanation of this project, we have found that ISIS is a powerful tool to simulate UAS due to its flexibility to include mission-specific services. In addition, the difficulty flight testing due to the current legislation of the air space, to the costs of deploying the mission and UAS risk of accidents give more importance on creation of this simulation platform.

As we have seen, ISIS to simulate a specific mission UAS interacting with the environment, but what about the air traffic and airspace of the legislation? Thanks to the eDEP integration in ISIS that we are implemented, now we can simulate the missions that a UAS has to interact with air traffic. But this is only the beginning, because due to the interoperability offered by ISIS will soon extend to other simulation platforms in which we can extract his air traffic.

We have accomplished all objectives the proposed at the beginning of this document:

- Homogenize data between the two simulation platforms. We have integrated two platforms that are programed in two different programming languages. ISIS only reads the telemetry data format composed by position, speed, bearing and Callsign (Register). We have known to traduce ADSB messages generated by eDEP in the format that ISIS could understand.
- To group the air traffic from the ATM simulator to UAS simulator in a real time. As we have accomplished homogenize data, thanks to the Traffic Service designed by us, now we can extract the air traffic from eDEP and include them in ISIS simulations.
- Include the UAS in the air traffic simulator as an aircraft more. The knowledge obtained of eDEP environment during this project, allows to us manipulate data from ISIS to include our UAS simulations in the ATM simulation platform.
- To Design and test simulations for a UAS interacting with air traffic. As we seen in the chapter 5 called Simulations, we have designed eDEP scenarios with air traffic. This air traffic has been included in ISIS allowing to see the UAS simulation interacting with them.

Remember that eDEP not only can generate traffic, but in addition we can control it. So, for example we can test for air traffic management in a classroom with several computers for students while one or several UAS (simulated from ISIS) are under their command.

In short, an application of integration between ISIS and eDEP opens up many possibilities.

Proposals for improvement

1- When creating a simulation, it is necessary to open and modify files one at a time of eDEP "airspace.dat", "airtraffic.dat" and "demo.gsdk." There could be a user-friendly application in which eDEP modify files in one display.

2- Should be able to unify the way that opens ISIS services. Each time you execute a service cover the vision of the console having to minimize the display of the service.

3- Should increase the frequencies at which ADSB messages are transmit because during the execution of integration and eDEP ISIS, airplanes suffer stoppages.

REFERENCES

[1] The Intelligent Communications and Avionics for Robust Unmanned Aerial Systems group , ICARUS researchers group.
http://icarus.upc.es/

[2] Royo Chic, P., “ICARUS Simulation Integrated Scenario (ISIS)”, Chapter 4. In An Open Architecture for the Integration of UAS Civil Applications, May 2010.

[3] Royo Chic, P., “Introduction”, Chapter 1. In An Open Architecture for the Integration of UAS Civil Applications, May 2010.

[4] Royo Chic, P., “Previous Work”, Chapter 2. In An Open Architecture for the Integration of UAS Civil Applications, May 2010.

[5] Barrado,C. and Lopez J., “MAREA user manual” (June 29, 2009)

[6] eDEP, EUROCONTROL Projects. eDEP description
http://www.eurocontrol.int/eec/public/standard_page/ERS_edep.html

[7] ADS-B. ADS-B description
http://en.wikipedia.org/wiki/Automatic_dependent_surveillance-broadcast

[8] ADS-B Technologies. Figure ADS-B
http://www.ads-b.com/default.htm

[9] COMSOFT. ASTERIX products. ASTERIX Standard description
www.comsoft.aero/download/atc/product/spanish/asterix_spanish.pdf

[10]ASTERIX, EUROCONTROL CAT021
http://www.eurocontrol.int/asterix/public/standard_page/documents_archives.html