A Highly-Automated RPAS Mission Manager for Integrated Airspace

Conference Paper · September 2015

DOI: 10.1145/2899361.2899363

4 authors, including:

Hector Usach Molina
Universitat Politècnica de València
7 PUBLICATIONS **1** CITATION
SEE PROFILE

Alfons Crespo
Universitat Politècnica de València
236 PUBLICATIONS **1,609** CITATIONS
SEE PROFILE

Pedro Yuste
Universitat Politècnica de València
32 PUBLICATIONS **158** CITATIONS
SEE PROFILE

A highly-automated RPAS mission manager for integrated airspace

H. Usach Molina
Instituto de Automática e Informática Industrial
Camí de Vera, s/n
46022 Valencia, Spain
hecusmo@doctor.upv.es

J. Vila Carbó
Universitat Politècnica de València
Camí de Vera, s/n
46022 Valencia, Spain
jvila@upvnet.upv.es

A. Crespo Lorente
Instituto de Automática e Informática Industrial
Camí de Vera, s/n
46022 Valencia, Spain
acrespo@disca.upv.es

P. Yuste Pérez
Universitat Politècnica de València
Camí de Vera, s/n
46022 Valencia, Spain
pyuste@disca.upv.es

ABSTRACT

This paper addresses the problem of defining Mission Plans for Remotely Piloted Aircraft Systems (RPAS) and designing a software architecture for executing such plans. A RPAS Mission Plan for integrated airspace usually includes flight procedures in controlled airspace as well as procedures for the operations area, which usually is in a non-controlled airspace. Procedures for controlled airspaces must adhere to standard regulations, but nothing has been yet standardised for the operation in non-controlled airspaces. Some coherent extension to traditional Flight Plans is needed to specify all flight procedures in a coherent manner. This paper advocates extending ARINC-424 route definitions with RPAS specific features. On the other hand, RPAS do need a higher level of automation than manned aircraft due to many reasons: lower situational awareness, need for higher autonomy upon C2 link failures, or higher accuracy operation. This fact has important implications on the degree of automation and the definition of the Mission Plan. Firstly, different operational modes with different levels of automation are required. Secondly, traditional Flight Plans should be also extended to specify automatic flight procedures triggered by events like C2 link failures, or collision avoidance that could lead to unsafe conditions. The paper also discusses in detail the design of a Mission Manager for supporting the previous features based on a three-tiered (3T) architecture. The porting process of this architecture to an ARINC-653 compliant execution platform based on XtratuM is also introduced.

ATACCS'15 September 30–October 2, 2015, Toulouse, France.

CCS Concepts

•**Applied computing** → **Aerospace;**

Keywords

RPAS, Mission manager, Mission plan, Automatic flight procedures, Software architectures

1. INTRODUCTION

The process of insertion of Remotely Piloted Aircraft Systems (RPAS) in integrated airspace is still under discussion. However, experts agree on the fact that it must be linked to the European ATM Master Plan and the ICAO Global Plan/ASBU timeline [13]. This implies that RPAS has to fit into the ATM system transparently, with no additional hazards to existing operations. To this end, several technological defies arise, such as the command and control (C2) link, the detect and avoid capability, the communication architectures with ATC, or the contingency procedures.

RPAS require a higher level of automation than manned aircraft. This is mainly due to a lower situational awareness in RPAS with current interfaces for remote control: pilots cannot hear the engine, they cannot feel vibrations, acceleration or motion and they lose stereoscopic and peripheral vision. Most losses of unmanned aircraft have been attributed to human error and a limited ability to detect and avoid trouble. Another important reason is unreliable communications links [29].

Automated navigation and control systems offer increased accuracy in executing complex procedures when compared to remote human operation. They also offer the benefit that they do not suffer from fatigue. On the other hand, there is also a need for reducing dependency on a reliable remote control.

The degree of automation in RPAS is still an open issue. ICAO Manual on RPAS (Doc.10019, AN/507) [10] distinguishes different operational configuration for the RPS (Remote Pilot Station): *BVLOS Category A – Direct Control* provides the greatest level of remote pilot control of the RPA, allowing inputs equivalent to a control stick. *BV-*

LOS Category B – Autopilot Control provides less control of the RPA, still allowing speed, altitude, heading and vertical speed to be controlled. *BVLOS Category C – Waypoint control* provides limited control by the remote pilot of the RPA during flight. The flight planned route can only be altered through waypoint entries. Automatic take-off and landing are considered an "operational specificity" by the document and they are abilities that must be subject to controller training and sometimes restricted by aerodrome operators. In this paper, we even consider to introduce some degree of automation (operational specificities) that go beyond "Category C", such as object or mid-air collision avoidance. The ICAO document also provides a discussion of lost C2 link contingency options. Some of them are highly automatic and could result in the RPA flying "not under command" (automatically) for long duration.

From the above discussion, it can be deduced that a key point of a RPAS Mission Manager is providing several operational configurations switchable at runtime either manually by the operator or automatically.

Another key point is the definition of Mission Plans. In RPAS it is desirable to anticipate situations requiring a reactive behaviour upon detecting a safety related event that triggers an avoidance procedure, a mission replanning, a waiting procedure, etc. It is convenient to specify these behaviours in the Mission Plan. The system should also be flexible for specifying new automatic flight procedures that can be triggered by safety related events. The alternatives for defining RPAS mission plans are also an open issue and most of the proposal are system dependent [2, 21, 22, 26]. However, our point of view is that flying in integrated airspace should comply with existing proposals for RNAV/RNP navigation.

Designing an architecture for fulfilling all previous requirements is a difficult issue. In our experience, the 3T (three-tiered) software architecture is maybe one of the most well founded proposals to solve this problem. It has been in development in several forms since the late 80's as a result of the failure of the "sense-plan-act" architectures on real-world robots [5, 8]. The 3T version of this architecture has been used by NASA since 1992 for different applications. It has been also adopted for some onboard mission management of RPAS [1, 2, 18] and in the Mars explorer mission [11].

The rest of the paper is organised as follows: Section 2 presents the design of the proposed Mission Manager, including operational modes, Mission Plan properties and specification, and the interface with other avionics applications. Section 3 describes the system implementation. Section 4 analyses the development process from the design to the deployment on the target platform. Section 5 presents a Mission Plan demonstration in a simulation environment. Finally, Section 6 outlines some concluding remarks.

2. MISSION MANAGER DESIGN

This paper describes the design of a Mission Manager for a RPAS flying in integrated airspace. Flying in integrated airspace has many implications on the design. To start with, the Mission Manager should be able to make use of standard RNAV/RNP route specifications based on ARINC-424 path terminators. In addition, these specifications must be extended to include RPAS specific flight procedures in a flexible way. Such operations include maneuvers like orbiting around a point, scanning patterns or payload related commands. Finally, *performance monitoring* must be properly measured and handled in order to comply with Performance-Based Navigation (PBN) accuracy specifications.

The level of automation of a RPAS Mission Planner and the human factors that come into play when decision acts must be taken by a human operator interacting with a highly-automated planner are one of the most interesting fields that are still open and require further research [17]. The seminal work by [24] first addressed the levels of automation and introduced a detailed taxonomy of such levels. But RPAS have introduced different and greater human factors challenges. These arise primarily from the fact that operator and aircraft are not co-located. These factors can be summarised in two key issues: a loss of situational awareness, and C2 link communication problems. Special attention deserve *contingency procedures*. They refer to the actions to deal with failures of a safety-critical control system. In the case of RPAS these actions are established by aviation authorities and the software design has to be approved by certifying authorities. Most of the times there exist several contingency options. For example the ICAO RPAS Manual establishes five contingency options for C2 link loss that the RPAS operator has to consider. Decisions regarding the chosen option may be different depending on some factors, specially the segment of flight where the failure occurs. An important point here is the degree of automation in the *decision acts*: in some cases the automatic system can request the operator to select a procedure, in others it may only request his approval, and in some others, as in the case of the C2 link loss, the operator has no choice to specify any decision in real-time, and it has to be completely automatised.

The effectiveness of a contingency procedure depends on the following key points: the contingency procedure is technically correct and adequate to deal with the situation; the decision act upon a failure occurrence results in choosing an adequate plan; and finally, no execution errors are committed by the operator. Human factors play an important role here. Wrong *decision acts* have been reported as one of the main causes that may lead to an unsafe situation [4]. The goal of automation is avoiding this, but the allocation of responsibilities between the automatic system and the operator is sometimes difficult. *Execution errors* of a chosen plan can also lead to unsafe situations, even if the plan was previously trained. There are multiple reasons for such errors: insufficient or degraded sensory input, lack of information, C2 link delays, or an inadequate design of the human-machine interface, as in the case of [12]. Despite their importance, human factors go beyond the scope of this paper.

Another important point in the design of a RPAS Mission Manager is providing several operational modes with different levels of automation. This will be described next.

2.1 Operational modes

The proposed Mission Manager roughly offers all operational configurations defined by ICAO for the Remote Pilot Station (RPS) extended with modes to deal with automatic contingency procedures:

- *Manual mode* – The aircraft is controlled by the remote pilot allowing inputs equivalent to a control stick. It is the lowest level of automation, requires a lot of expertise and it is not usually used or even trained by pilots in some large RPAS.

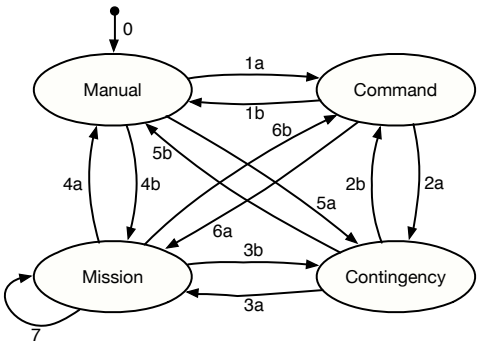

Figure 1: Operational modes and transitions in the Mission Manager automata.

- *Command mode* – It basically provides the typical functions of an autopilot (Mode Control Panel) and it allows to pilot the RPAS setting the following parameters: heading, altitude, speed, and vertical speed/vertical flight path angle. It is the most common way of piloting large RPAS.
- *Mission mode* – It is the highest level of automation. The RPAS is guided using the route definition of the Mission Plan in the vertical and lateral profiles in an automatic way. ARINC-424 path terminators are used to define the guidance in this mode. The Mission Plan can be changed by a *replanning* operation.
- *Contingency mode* – It allows to specify automatic procedures triggered by an event which usually correspond to a *safety condition*, as avoiding a mid-air collision or an obstacle, or a *system failure* as a C2 link loss of connectivity.

The *Mission* and *Contingency* modes are actually macro-modes which can include some other submodes. Submodes are associated with specific flight procedures. For example, the *Mission mode* can include submodes for defining automatic *en-route*, *take-off*, and *landing* procedures, whilst the *Contingency mode* includes submodes for dealing with events that could potentially lead to unsafe situation. The latter includes a *holding* procedure that makes the RPAS start loitering, a *NFZ avoidance* procedure which performs a maneuver to bypass a no-fly zone (NFZ), and a *collision avoidance* procedure to automatically avoid mid-air collisions. Nonetheless, the most common contingency procedure consists of evolving unconditionally to *Manual mode* or *Command mode*.

Concerning design issues, the system is modeled as an event-triggered state machine shown in Figure 1. Modes and transitions are specified using an automata defined by a *transition table*. It is important to keep this automata as much simple as possible so the RPAS has a predictable behaviour that can be easily certified. Another key point is flexibility for defining and detecting new event types and for defining new mission and contingency procedures.

To this end, the automata only requires to specify the events that trigger transitions to other operational modes (macro-modes). Then, each macro-mode manages its own inner transition table which triggers the appropriate procedure. In this way, some important and difficult details are not provided at the main level; for example, which kind of collision avoidance procedure will be used (climb, descend, horizontal) or which waypoint in the Mission Plan will the contingency procedure return after the avoidance procedure. This task is assumed by the designer so several implementations are possible.

Furthermore, the interface of a contingency procedure must specify the definition of the events that make the system to evolve from *Contingency mode* to some other operational mode (*Manual*, *Command*, *Mission*). For example, after a collision avoidance procedure the system may return to course in *Mission mode*. Sometimes, the Mission Plan used by the *Mission mode* can be changed by a *replanning* operation. For example, after a C2 link failure procedure the system may execute a *go home* procedure towards the nearest landing site –as proposed in [10]–, which consists of a *replanning* operation and then going back to *Mission mode*.

2.2 The Mission Plan

A RPAS mission flying in integrated airspace typically goes through one or several flight phases flying in controlled airspace (airways, TMAs) to/from a working area, and some other flight phases flying into the working area which is usually a non-controlled or segregated airspace. This is usually the flight phase in which the payload comes into action, although the working area could also be located in some class of controlled airspace. In either case, the flight plan for the controlled airspace should be compatible with airspace regulations, should be approved by aviation authorities, and should contain more or less standard flight procedures that ATC can easily interpret and control. However, flight phases in non-controlled airspace do not need to adhere to any standard procedure, can be RPAS specific, and should contain payload specific commands.

The Mission Plan for the integrated airspace should be preferably based on ARINC-424 path terminators (CA, CF, DF, FA, IF, etc) to comply with RNAV or PBN regulations. The Mission Plan for flight phases in non-controlled airspace probably cannot be expressed in terms of path terminators but our proposal is to extend it with RPAS specific commands like ORBIT, SCAN, or LAND. The specification of these extensions should be syntactically equivalent to the one of Table 1 for path terminators. That requires the ability to define new *primitive procedures* for the guiding system.

Finally, a RPAS Mission Plan should be also extended to specify all automatic or semi-automatic procedures that have been considered necessary after carefully analysing safety implications and human factors involved. This is mostly the case of contingency procedures and it implies being able to specify an state automata. Up to now the automata was hardcoded and there was no flexibility in specifying or changing the contingency procedure. This extension supposes a major syntactical change in the specification because it requires declaring an automata with events that trigger transitions. This specification allows to state which contingency procedure will be triggered by an event and which operational procedure will be fired after the contingency procedure is executed. Events have to be defined as predicates of some state variables that have to be made visible to the Mission Manager. Events are raised as a result of the continuous monitoring of these state variables. Some events may have an internal origin, for example NAV_ERROR is raised when some Mission Plan specification cannot be ac-

Table 1: Primitive procedures and definition parameters

Behaviour	fix1lat	fix1lon	fix2lat	fix2lon	altitude1	altitude2	lim speed	course	fly-over	radius	turn dir	end type	lim value
CA							O	✓				3	✓
CF	✓	✓			O	O	O	✓					
DF	✓	✓			O	O	O						
FA	✓	✓					O	✓				3	✓
IF	✓	✓			✓	O	O	O					
RF	✓	✓			O	O	O			✓	✓		
TF	✓	✓			O	O	O		O				
VA							O	2				3	✓
VI	1	1			O	O	O	2				4	✓
FLY TO	✓	✓			O	O	O	O					
ORBIT	✓	✓					O			✓	✓	5	✓
SCAN	✓	✓	✓	✓	O		O	6					
LAND	7	7			8		O	9					

✓– Required
O – Optional
1 – Reference coordinates
2 – Heading not course
3 – Altitude type
4 – Bearing type
5 – Altitude, time, or laps type
6 – Initial course
7 – Runway threshold coordinates
8 – Airport elevation
9 – Runway heading
Shaded – Not applicable

complished with the required accuracy. Some other events have an external origin as RA_ADVISORY upon an ACAS advisory or C2_FAILURE upon a C2 link failure detection.

Section 5 shows an example of a Mission Plan and how the Mission Manager executes it.

Table 1 shows the currently implemented primitive procedures along with their definition parameters. It comprises 9 out of the 14 path terminators allowed in RNAV operations [9], including initial fix (IF), track to fix (TF), direct to fix (DF), course to fix (CF), radius to fix (RF), fix to an altitude (FA), course to an altitude (CA), heading to an altitude (VA), and heading to intercept (VI).

In addition there are some RPAS specific procedures, as FLY TO, ORBIT, SCAN and LAND. FLY TO is used to navigate towards a waypoint; ORBIT is used to orbit around a point until it reaches an altitude, an elapsed time condition, or a number of laps condition; SCAN is used to fly a scanning pattern as described in [7]; finally, LAND is used for flying a glide-slope towards a runway. Nevertheless, this list is being updated so new functionalities are being added to Mission Plans.

Some procedures in Table 1 have optional parameters that, if used, can provide different control modes and different behaviours. For example, the *altitude1* and *altitude2* fields; if set, they provide automatic control in the vertical profile to achieve a given altitude window (at or above *altitude1*, and/or below *altitude2*); if not, the altitude is maintained or specified in the Mode Control Panel. Another example is the *course* field in the FLY TO procedure; if set, it performs a Dubins path [14] to intercept the fix at the commanded course; otherwise, a direct course to the fix is followed. The important point here is that the resulting trajectory is computed at run-time. Another important field is the *end type* which identifies the type of termination condition checked at run-time to asses if the procedure goal has been successfully reached. It is used in the CA, FA, VA, VI, and ORBIT procedures to check different conditions: reaching an altitude, bearing to a point, number of laps performed or elapsed time. The target value for that condition is specified in the *lim value* field. For example, if an ORBIT is defined with an *end type* field equal to *altitude* and a *lim value* of 3000, the aircraft will orbit around (*fix1lat*, *fix1lon*) until it is at 3000 ft AMSL.

Regarding the formal definition of Mission Plan, it is currently implemented as a matrix, in which each row defines a mission procedure with columns being the parameters. Thus, the Mission Plan results in a $m \times n$ matrix, where m is mission dependent and n is fixed according to our implementation (the number of columns in Table 1).

2.3 System Structure

The *Mission Manager* is one of the components of the *Flight Management System* (FMS) of the RPAS. This system is schematised in Figure 2. In this way, it interfaces upstream with the *Navigation System*, which monitors the aircraft position and attitude and is designed to accomplish PBN [27]. Its main input is the Mission Plan, and it may have additional inputs from other sources that refer to other environment information, such as traffics, weather, terrain, etc. The more information sources are available, the more autonomous the system may become, and the higher the system integrity is.

On the other side, the guidance system interfaces downstream with the *Flight Control System* (FCS), an autopilot developed in our research group and described in [6]. The main goal of the *Mission Manager* is to provide the reference values for all the controlled variables.

3. MISSION MANAGER ARCHITECTURE

The Mission Manager is designed as a layered architec-

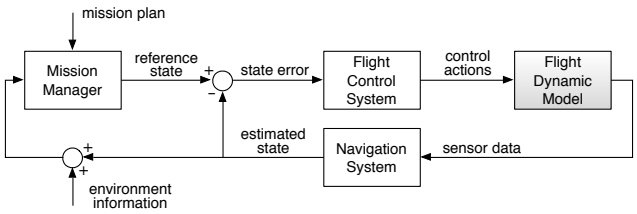

Figure 2: Flight Management System schema.

ture also called 3T architecture with each layer providing a different level of abstraction on the guidance process. This way, the Mission Plan procedures, hereinafter *complex behaviours*, are decomposed into a sequence of elementary maneuvers that can be directly interpreted by the FCS. The main advantage of this architecture is that it allows to enhance the system functionality by adding new behaviours at a reduced time and effort.

The overall system architecture is schematised in Figure 3. It comprises the *deliberate layer*, the *sequencing layer*, and the *reactive layer*, which are described next.

The reactive layer

The base layer, called reactive layer, contains the set of possible elementary maneuvers which are the core of complex behaviours. They can be thought as the basic skills or capabilities of the aircraft, hence they are closely linked to the FCS. In fact, the elementary maneuvers are used by a guidance algorithm to generate the target value for the controlled variables. The basic maneuvers can be divided into vertical profile maneuvers, or VNAV, and lateral profile maneuvers, or LNAV. They can be further classified by the type of action required to perform the maneuver as *attitude* or *path* maneuvers. Table 2 gathers the set of elementary maneuvers according to these criteria.

Table 2: Elementary maneuvers in the reactive layer

	Attitude	Path
LNAV	HEADING ROLL	TRACK CIRCLE LOC
VNAV	VSPEED	ALTITUDE VPATH G/S

Each basic maneuver is defined by a different number of parameters. For example, a TRACK maneuver is defined by a To/From indication, the coordinates of a fix, and the desired track to/from that fix; a CIRCLE maneuver is setup with the coordinates of the turn center, the turn radius, and the turn direction; but a HEADING maneuver or a ROLL maneuver just need the heading reference or the roll reference for that leg, respectively. The LOC (for localiser) and G/S (for glide-slope) maneuvers are used to describe the path along the glide path in a landing procedure; and the VSPEED and VPATH maneuvers define climbs or descents with constant rate-of-climb or vertical flight path angles, respectively.

It is important to consider that only path-based maneuvers need the previously mentioned guidance algorithm to generate reference values for the controlled variables. As the definition parameters of the attitude-based maneuvers contain all the information needed by the FCS, they can be

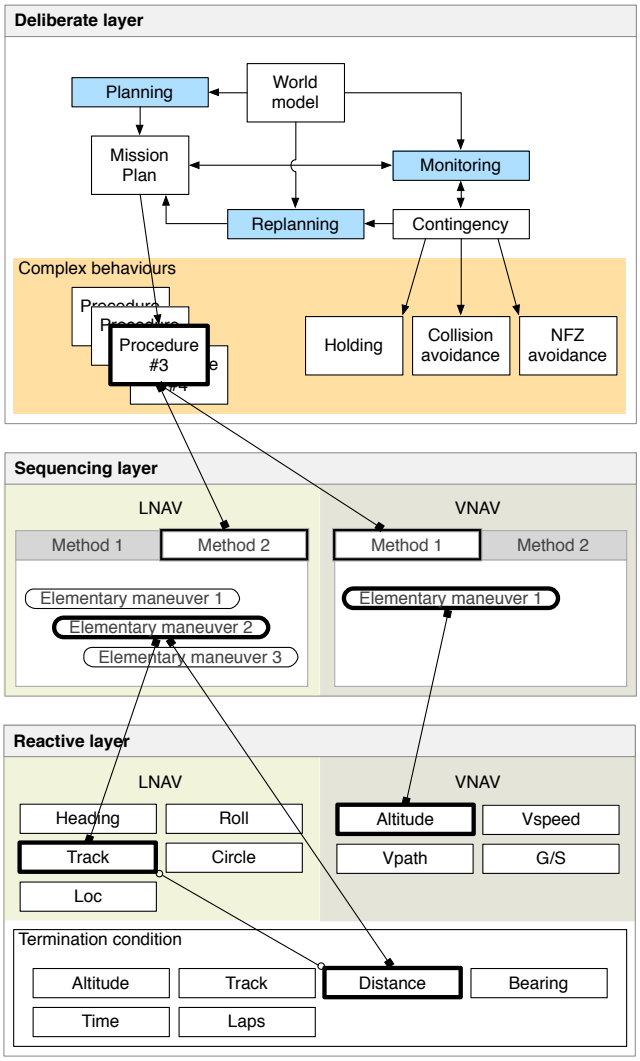

Figure 3: 3T architecture. In bold, an example of active states at each layer and their relations.

passed directly to the FCS interface. However, path-based maneuvers need to be preprocessed as described in previous works [7, 28].

Besides the definition parameters, a maneuver may be linked to one or more termination conditions, as shown in Figure 3. A termination condition checks whether a certain relation between the state of the aircraft and the maneuver itself is met or not. If so, the current skill is ended and an event is raised so that the higher layers of the architecture handle it. In this context, a termination condition could be reaching an altitude, the distance or the bearing to a point, etc.

The sequencing layer

The sequencing layer works as a bridge between the deliberate layer and the reactive layer. Its function is to split high level of abstraction operations or complex behaviours requested by the deliberate layer into a sequence of elementary maneuvers that can be interpreted by the FCS to comply with the objectives of the mission. Complex be-

haviours allow the aircraft to perform advanced navigation tasks. These include path terminators, RPAS specific procedures, but also mid-air collisions avoidance, self-separation from other traffics, or flying back home in case of certain contingencies.

Thus, complex behaviours are made up by one or more elementary maneuvers, so complex maneuvers are splitted into elementary maneuvers that are queued in a sequencer. Actually, the LNAV and VNAV profiles have two independent sequencers, both running concurrently. The system must ensure that each sequencer activates one and only one maneuver at a time, preventing from inconsistent states. Furthermore, the resulting sequence of elementary maneuvers may not be unique. A certain complex behaviour may contain a list of *methods* for complying with the mission specification (see Figure 3). A method is defined as a sequence of elementary maneuvers, together with an activation condition that is checked at run-time [8] . For example, the activation condition can be the initial bearing or distance to a point, or the next Mission Plan procedure. In this way, it is possible to execute a complex behaviour through several methods. The selected one tries to optimise the trajectory and the leg adaptation.

A large number of different complex behaviours can be defined using elementary maneuvers. For example, a TF is implemented as a single TRACK maneuver in LNAV, and optionally a VSPEED + ALTITUDE maneuver in VNAV. A CF consists of an intercept leg, in which the aircraft is conducted towards the closest point over the intended course, and a second leg flying the corresponding course. In this case, the intercept leg is implemented as a Dubins path using the sequence CIRCLE + TRACK + CIRCLE so that the aircraft intercepts the leg with an appropriate track, and the final leg is performed through a TRACK maneuver.

RPAS-based operations are also implemented in this way. The LNAV sequencer in a FLY TO behaviour contains two methods: one executing a single TRACK towards the target waypoint when the 'course' field from Table 1 is not specified; and other executing a Dubins path. Or the SCAN behaviour, which defines a scanning pattern as a sequence of $[\text{TRACK} + \text{CIRCLE}]^n$ + TRACK maneuvers, meaning that the first two steps are repeated according to the planning algorithm.

The deliberate layer

The deliberate layer is at the top of the architecture and thus it is responsible for coordinating the execution of the Mission Plan. Since it is the intelligent component of the system, it *deliberates* about how to achieve the mission goals taking into account the system constraints. In this context, this means that it is responsible for managing the mission, what involves planning, monitoring, and replanning functions.

The *planning* function is executed offline to analyse the feasibility of the Mission Plan. It checks that the different procedures that comprise the plan have an *a priori* reachable goal with some given aircraft performance and resources. That is done checking conditions such as distance, turns, or altitude constraints, or even the sequence of procedures itself. However, it does not take account of ATM aspects such as airspace capacity, NOTAM or METAR information. The output of the planning function is a static list of complex behaviours consistent with the aircraft performance.

At mission execution, the deliberate layer activates each complex behaviour from the Mission Plan and passes their definition parameters to the sequencing layer, where the appropriate method at each sequencer is selected.

This level of the architecture is also responsible for online event *monitoring*. This consists of listening and handling different events that may cause changing the operational mode of the Mission Manager automata. During mission execution, integrity tests are also performed. As a result, some events can be raised. Some of them are related to navigation. They occur when a goal of the Mission Plan cannot be reached with the current state of the aircraft. A typical example of this situation occurs when some procedure imposes an altitude goal in a VNAV maneuver which cannot be achieved with some given speed restriction and the aircraft performance. Some other events are related to the environment (weather conditions, other traffics, NFZ, C2 link, etc.), alerts or system malfunctions. Different addons can be connected to this layer to access environmental information ('World model" in Figure 3) and supplying the required data. The traffic awareness can be obtained plugging a transponder model that provides position, speed and track data from other aircrafts. It is also possible to include airways, navaids, or airspace considerations –such as the Required Navigation Performance (RNP)– by using a navigation database.

The deliberative layer must react and handle these events changing to the *Contingency mode* or to the *Manual mode*. Handling an event is a preemptive action: it can interrupt the execution of any underlying sequence of tasks. *Manual mode* is entered in an unpredictable situation that cannot be automatised, such a combination of multiple events. *Contingency mode* deals with situations that can be safely automatised in a predictable way. Any problem in *Contingency mode* will final result in evolving to *Manual mode*. Executing a contingency procedure may involve, in some cases, a mission *replanning*. In this situation, the aircraft temporarily abandons the original Mission Plan and executes a contingency procedure. When this is complete, the deliberate layer must face the problem of where to resume the plan. In some cases, as for example, when a NFZ is avoided, it is a matter of selecting the next reachable waypoint in the original plan. Some other case involving flight performance limits or strong winds in the area may require to replace the original plan by a new one.

4. SYSTEM DEVELOPMENT PROCESS

As previously explained, the Mission Manager results in a complex system containing several operational modes. In each of them, the manager executes some procedures fired by the occurrence of certain events. It is important to take into account that the resulting system is to be embedded into an aerospace application which must satisfy rigorous development and verification standards. As a consequence, the software design methodology should be able to pass a certification process. This section briefly describes the development process: from system design to its deployment on the target platform.

4.1 Design methodology

Critical systems such as an aerospace application require a high level of integrity and thus it is crucial to carefully consider the system design from the point of view of the software design methodology. System design is strongly af-

fected by the certification process which basically consists of three phases: identifying the essential airworthiness requirements, providing the required documentation, and providing the means of evidence. The framework for airworthiness requirements in Spain is mostly influenced by the STANAG 4703 requirements [19]. Essential requirements include integrity, operational, and organisations. There is also a safety management plan. It is out of the scope of this paper to make a comprehensive list of all requirements that have an impact of the Mission Manager. They include aspects such as the different automatic operational modes (UL47.1) for each flight phase, the data link loss (UL26.3), the Safety Assesment (UL.30) or the software Development Assurance Levels (DAL) (UL.31). These requirements rather point out a problem than require a particular solution. This solution has to be agreed or negotiated with the Certifying Authority. This is where Software Design Methodologies, like DO-178, play a key role since verifying or providing evidences for the different requirements can be easier done at the different system levels. The means of evidence include the Software/Hardware architecture and DAL allocation.

However, although all these details have been taken into account in the design methododology, it is worth saying that we have not dealt still with the certification process.

In previous works [7, 28], we advocate the use of a Model-Based Design (MBD) methodology to help satisfying the objectives of DO-178C, one of the most extended certification standards for airborne systems. We also considered an interest goal using an execution platform that complies with the ARINC-653 standard for Integrated Modular Avionics (IMA) architectures.

The development process of MBD roughly consists of the following steps: a) System modelling, b) Model design and implementation, c) Model validation, and d) Deployment, which can be mapped to the DO-178C workflow.

In our experience, MBD offers several advantages throughout all the design process. It permits reducing the system complexity by dividing it into a set of hierarchical models. It also allows to start the validating process at early steps of the design, thus reducing development time and costs. In addition, automatic code generation for the target system can be used, thus easing this process and producing free-error code.

Regarding software requirements, they can be roughly classified into functional, performance, and operational requirements. In general, functional requirements specify that the mission manager will sequence the Mission Plan procedures, guaranteeing the overall system integrity. Performance requirements mainly describe that the system shall comply with ICAO's rules of the air and the A-RNP navigation specification. Finally, operational requirements determine that system shall be designed in a modular way so that new components or functionalities can be easily introduced.

4.2 Deployment

As previously stated, one of the benefits of MBD is that it allows automatic code generation for the target system. This is possible by prototyping the system in Matlab and Simulink, and using the toolboxes *Simulink Coder* and *Embedded Coder* in conjunction with a series of tools for adapting the code generation. In this way, it is possible to easily deploy the designed model to a real-time execution platform.

In our case, we use a target based on XtratuM [15] running

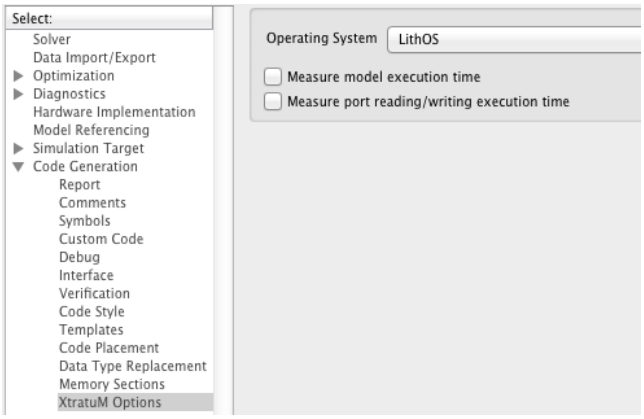

Figure 4: XtratuM configuration pane.

LithOS [16]. XtratuM is an hypervisor for safety-critical embedded systems developed at the Universitat Politècnica de València. With respect to the LithOS real-time operating system, it includes all the functionalities required to build systems based on ARINC-653, the standard interface for IMA architectures. IMA structures a system in a partitioned manner, which means that several applications running on the same target are isolated in the spatial and temporal points of view. Such an execution environment guarantees that possible faults are not propagated to other partitions.

Regarding the porting process from the Simulink design to the XtratuM environment, several issues need to be addressed. The most important ones are mainly related to the partitioning scheme. They include the communication mechanisms, the underlying operating system used in each partition , the concurrency model, etc. A Simulink tool for defining all these issues has been developed (see Figure 4).

More details about these issues are presented in previous works [7, 28].

5. A MISSION PLAN EXAMPLE

This section presents a Mission Plan example and how it is executed by Mission Manager. The scenario has been executed in a simulation environment based on the X-Plane flight simulator. It consists of the full application code, including the navigation and control partitions, running on XtratuM with a LithOS extended version which provides Linux services for UDP communication. The application is connected to a flight simulator which runs the flight dynamic model of the aircraft. The aircraft model used in this simulation is the Cessna 172-P. It has been selected because it has widely available technical data and flight models [3, 23, 25], needed to develop the control system and to consider the dynamic constraints.

The Mission Plan is presented in Table 3. It defines an observation mission around an area of ecological value near the city of Valencia (Spain) departing from the runway 18 of the ICAO's LECN airport (Castellón de la Plana). For brevity, the presented Mission Plan only covers some flight phases, omitting the return after the observation operation. Basically, the route consists of following the B-28 airway, a lower ATS route, from the SOPET waypoint to VLC using a TF. Then, the aircraft is intended to leave this airway to overfly the zone of interest defining a SCAN operation.

Table 3: Mission definition example

Behaviour	fix1lat	fix1lon	fix2lat	fix2lon	altitude1	altitude2	lim speed	course	fly-over	radius	turn dir	end type	lim value
IF	39.9325	0.0261			5000	–	–	–					
FLY TO	39.8339	-0.0047			–	–	–	228					
TF	39.4856	-0.4833			–	–	–		Y				
SCAN	39.3500	-0.4083	39.3167	-0.3000	2000		100	160					

Initially, the IF is located 4 NM beyond the airport, and the airway is intercepted using a FLY TO operation with a course specification. So finally the mission results in a combination of ATS procedures and RPAS operations.

Additionally, a potentially colliding traffic has been introduced flying the same route in the opposite direction. This is to simulate a traffic alert and the subsequent resolution advisory and mission replanning. The traffic data corresponds to a real traffic track with hex code 0200ED, recorded on 05-Nov-2014 using an ADS-B antenna located at our university. This traffic was following the upper route UM-445 and UM-985, going through waypoints BEGOX, VLC, and SOPET, at FL380 and flying at 500 mph. A mid-air collision alert is forced by matching the altitude from both traffics, resulting in a "head-on" conflict (since airways B-28 and UM-985 overlap although B-28 is in the lower airspace, and UM-985 in the upper). It could be argued that the scenario is not completely realistic, as the mission enters the TMA of Valencia and traffics are forced to fly at the same altitude in opposite directions, but it has been designed to serve as an example to illustrate the capabilities of the Mission Manager.

This data is supplied to our FMS application by a transponder model and processed by an ACAS model that produces a resolution advisory (RA). The resolution is solved automatically by a contingency procedure using a non-collaborative maneuver based on the CPA concept [20]. The implementation of this ACAS model is out of the scope of this paper. It is only worth noting that the ACAS subsystem is considered a source of environmental information ("World model" box in Figure 3). The information provided by the ACAS to the Mission Manager is the event RA_ADVISORY indicating the RA, the heading needed to avoid the collision, the event RA_TERMINATE indicating that the traffic advisory is over, and the heading needed to intercept the former Mission Plan.

The simulation output is presented in Figure 5. It shows both the planned and the actual mission execution of the host aircraft, under the control of the Mission Manager, and the path of the recorded traffic 0200ED. It can be seen how the Manager automatically takes the Cessna through the airway. When the ACAS system produces a RA_ADVISORY event, the Mission Manager reacts and switches to *Contingency mode* to avoid the mid-air collision, thus resulting in getting away from the intended Mission Plan. Then, the Mission Plan is resumed and, finally the SCAN procedure is flown.

Figure 6 shows in detail the execution of the contingency procedure. The behavior has two phases: avoiding the collision and recovering the trajectory defined by the Mission Plan. The figure shows the position of both traffics in the CPA which really occurs after the contingency procedure ac-

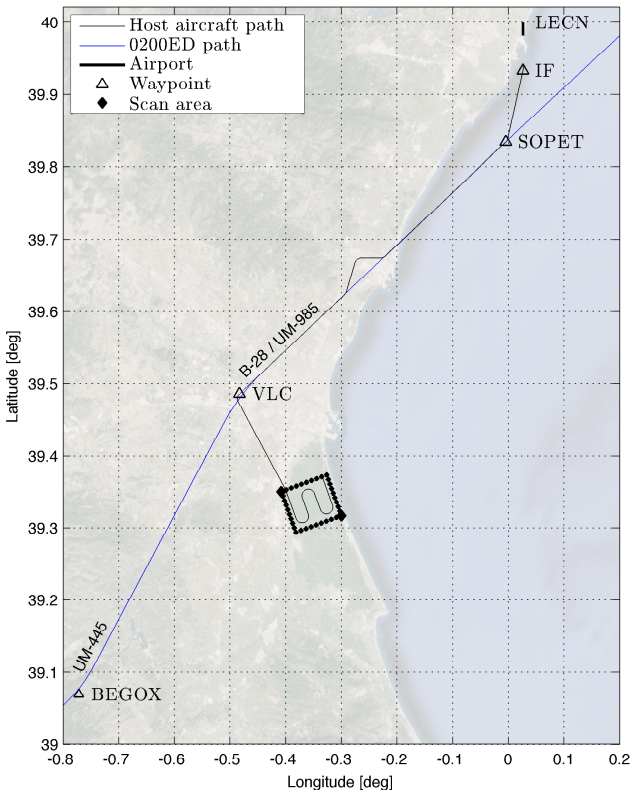

Figure 5: Mission Plan and simulation output.

tivation. The resulting paths from both the evasive and the recovery stages are also plotted in solid and dashed lines, respectively. Finally, the contingency procedure ends when the airway is intersected again.

Next, a description of how the 3T architecture manages the previous collision avoidance is presented. Keep in mind the schema of Figure 3 to follow this explanation.

After the RA_ADVISORY, the deliberate layer activates the *Contingency mode* . If the ACAS system decides to perform a horizontal evasive maneuver, the deliberate layer selects the HCOLLISION AVOIDANCE behaviour and passes it to the sequencing layer, along with its definition parameters. This behaviour is defined by three parameters: the heading reference for avoiding the collision, the heading reference for recovering the Mission Plan after avoiding the collision, and the maximum roll angle allowed. The first two parameters are computed by the ACAS model, while the third is supplied by the deliberate layer itself, since it is responsible for

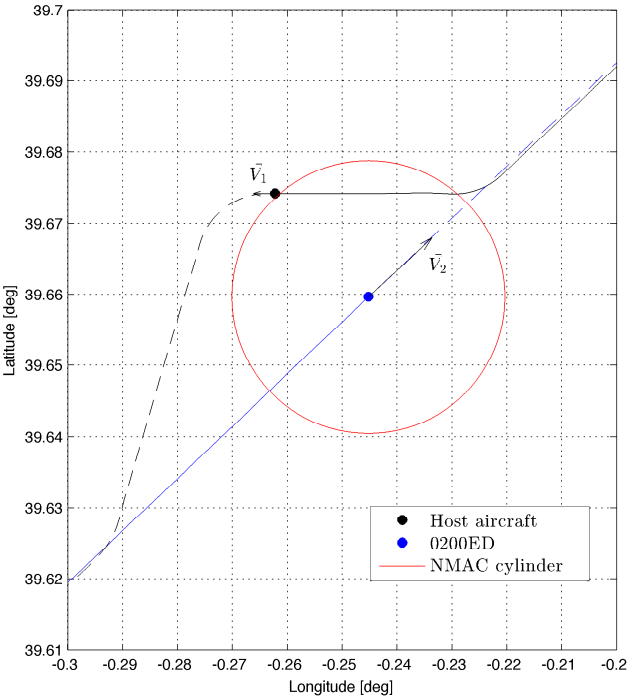

Figure 6: Collision avoidance procedure. Instant when CPA occurs.

taking into account the system constraints.

Then, the sequencing layer decomposes the complex behaviour in a list of four elementary maneuvers in LNAV, and only one in VNAV. The maneuver in the VNAV system consists of activating the ALTITUDE skill during the entire procedure, since it is a 2D conflict resolution maneuver. With respect to the LNAV sequence, the first two elements on the list (ROLL + HEADING) account for the evasion maneuver, conducting the aircraft out of the near mid-air collision (NMAC) cylinder, while the last two (ROLL + HEADING again) account for the recovery phase. Moreover, the previous elementary maneuvers are linked to a given termination condition. Both ROLL have a track termination condition, so the aircraft turns at maximum roll until it reaches the heading target. The first HEADING maneuver covers a special situation. It is not directly linked to any termination condition listed on Figure 3 but to the raising of an event: once the maneuver is activated, it is maintained until the ACAS sends a RA_TERMINATE event, meaning that the collision has been avoided, so the recovery phase can be started. Lastly, the second HEADING maneuver is linked to a bearing termination condition which indicates that the former plan has been recovered, so the COLLISION AVOIDANCE is over.

So when the list is planned, the first maneuver of each sequence is passed to the reactive layer. There, an event is sent back to the sequencing layer each time a termination condition is met, so that it activates the next elementary maneuver on the list. Once the list is completed, an event is sent back to the deliberate layer to recover the *Mission mode* from the next valid Mission Plan procedure.

6. CONCLUSIONS

The level of automation in RPAS is still an open issue and the need for a higher level of automation than in manned aircrafts has been outlined. This paper presents a highly-automated RPAS Mission Manager for integrated airspace. Increasing automation is accomplished through a wide system monitoring of events (failures, navigation errors, etc.) and an automatised and flexible handling of contingency procedures activated by those events. The ability to operate in non-segregated airspace has been achieved by defining Mission Plans as an extension to Flight Plans based on the ARINC-424 path terminator concept with RPAS specific procedures and contingency procedures. Up to now, most of the proposals for Mission Plans were system dependent.

An important issue in the design of the Mission Manager is the software architecture that supports all previous features, providing flexibility for introducing monitoring of new events and new automatised procedures. The proposed approach uses a three-tiered (3T) software architecture, where each layer provides a different level of abstraction. It offers several operational modes with different levels of automation, from manual to mission control, which are switchable in runtime either manually or automatically according to mission execution events. The resulting architecture provides a flexible environment for automating complex operations such as holding or scanning patterns, collision avoidance maneuvers, or go home procedures. This is done by decomposing the high level of abstraction operations into a sequence of elementary maneuvers that can be directly interpreted by any Flight Control System.

Finally, the process of porting this design to an Integrated Modular Avionics architecture based on XtratuM is presented. The experience has revealed that the porting process is feasible and easy to automatise. This can be achieved by mapping Simulink abstractions to the services of the ARINC-653 avionics standard interface. In this way, it is possible to prototype complex control systems in a simulation environment, to automatically produce the application code, and to deploy it to a real-time platform with reduced time and effort.

7. ACKNOWLEDGMENTS

This research has been financed by project GVA AICO/2015/126 (Ayudas para Grupos de Investigación Consolidables) of the Spanish Regional Government “Generalitat Valenciana”.

8. REFERENCES

[1] F. Adolf and F. Andert. *Onboard mission management for a VTOL UAV using sequence and supervisory control*. INTECH Open Access Publisher, 2010.

[2] F. Adolf and F. Thielecke. A sequence control system for onboard mission management of an unmanned helicopter. In *AIAA Infotech@Aerospace*, May 2007.

[3] J. Berndt and T. Peden. JSBSim open source flight dynamics model [online]. http://jsbsim.sourceforge.net. Accessed: 2012-02.

[4] K. Berry, M. Sawyer, and E. Austrian. Human factors assessment of RNAV approach and departure procedures. Technical report, FAA Human Factors Division (ANG-C1), 2013.

[5] R. Bonasso, R. Kerri, K. Jenks, and G. Johnson. Using the 3T architecture for tracking Shuttle RMS procedures. In *Proceedings of the IEEE International Joint Symposia on Intelligence and Systems*, pages 180–187. IEEE, 1998.

[6] B. Fons-Albert. Plataforma para diseño y ejecución de aplicaciones de aviónica. Master's thesis, Universitat Politècnica de València, September 2013.

[7] B. Fons-Albert, H. Usach-Molina, J. Vila-Carbó, and A. Crespo-Lorente. Development of Integrated Modular Avionics applications based on Simulink and XtratuM. In *Data Systems in Aerospace Conference 2013*. Eurospace, May 2013.

[8] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *AAAI*, volume 1992, pages 809–815, 1992.

[9] International Civil Aviation Organization. *Doc. 9613, AN/937: Performance-based Navigation (PBN) Manual*, 4th edition, 2013.

[10] International Civil Aviation Organization. *Doc. 10019, AN/507: Manual on Remotely Piloted Aircraft Systems (RPAS)*, 1st edition, 2015.

[11] C. Ippolito and G. Pisanich. Cognitive emotion layer architecture for intelligent UAV planning, behavior and control. In *Aerospace Conference*, pages 1–16. IEEE, 2005.

[12] C. W. Johnson. The hidden human factors in Unmanned Aerial Vehicles. In *26th International Conference on Systems Safety, Vancouver, Canada 2008*. International Systems Safety Society, 2008.

[13] D. Koehl. SESAR initiatives for RPAS integration. In *ICAO Remotely Piloted Aircraft Systems Symposium*, Montreal, March 2015.

[14] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[15] M. Masmano, I. Ripoll, A. Crespo, and J. Metge. XtratuM: a hypervisor for safety critical embedded systems. In *12th Real-Time Linux Workshop*, 2009.

[16] M. Masmano, Y. Valiente, P. Balbastre, I. Ripoll, A. Crespo, and J. Metge. LithOS: a ARINC-653 guest operating for XtratuM. In *12th Real-Time Linux Workshop*, Kenia, 2009.

[17] J. S. McCarley and C. D. Wickens. Human factors implications of UAVs in the national airspace. Technical Report AHFD-05-05/FAA-05-01, University of Illinois, Institute of Aviation, Aviation Human Factors Division, 2005.

[18] M. Niendorf, F.-M. Adolf, and T. Gerhard. Behavior-based onboard mission management for an unmanned fixed-wing aircraft. In *AIAA Infotech@Aerospace*, May 2012.

[19] North Atlantic Treaty Organization. *STANAG 4703: Light Unmanned Aircraft Systems Airworthiness Requirements*. NATO Standarization Agency, 2014.

[20] J.-W. Park, H.-D. Oh, and M.-J. Tahk. UAV collision avoidance based on geometric approach. In *SICE Annual Conference, 2008*, pages 2122–2126. IEEE, 2008.

[21] E. Pastor, P. Royo, E. Santamaria, M. P. Batlle, C. Barrado, and X. Prats. An architecture to automate UAS operations in non-segregated airspace. In *Proceedings of the 1st International Conference on Application and Theory of Automation in Command and Control Systems*, pages 5–14. IRIT Press, 2011.

[22] E. Pastor, E. Santamaria, P. Royo, J. Lopez, and C. Barrado. On the design of a UAS flight plan monitoring and edition system. In *Aerospace Conference*, pages 1–20. IEEE, 2010.

[23] J. Roskam. *Airplane flight dynamics and automatic flight controls*. Design, Analysis and Reseach Corporation, 3rd edition, 2001.

[24] T. B. Sheridan and W. L. Verplank. Human and computer control of undersea teleoperators. Technical report, Department of Mechanical Engineering, MIT, 1978.

[25] B. L. Stevens and F. L. Lewis. *Aircraft control and simulation*. John Wiley & Sons, Inc., 2nd edition, 2003.

[26] I. A. Troxel and A. D. George. Adaptable and autonomic mission manager for dependable aerospace computing. In *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 11–18. IEEE, 2006.

[27] H. Usach-Molina. Integridad y tolerancia a fallos en sistemas de aviónica. Master's thesis, Universitat Politècnica de València, September 2014.

[28] H. Usach-Molina, B. Fons-Albert, J. Vila-Carbó, and A. Crespo-Lorente. An autopilot testbed for IMA (Integrated Modular Avionics) architectures. In *Automatic Control in Aerospace*, volume 19, pages 435–440, 2013.

[29] C. Whitlock. When drones fall from the sky. *The Washington Post*, June 20, 2014.