An Architecture for the Development of Complex UAS Missions

E. Pastor, C. Barrado, P. Royo, J. Lopez and E. Santamaria
Dept. Computer Architecture, Technical University of Catalonia (UPC)
08860 Castelldefels, Spain
enric@ac.upc.edu

Abstract

The generalized development of UAS complex applications are still limited by the absence of systems that support the implementation of the actual mission. UAS design faces the development of specific systems to control their desired flight-profile, sensor activation/configuration along the flight, data storage and eventually its transmission to the ground control. All this elements may delay and increase the risk and cost of the project.

This paper introduces a flexible and reusable hardware/software architecture designed to facilitate the development of UAS-based complex applications. This flexibility is organized into a user-parameterizable UAS service abstraction layer (USAL). The USAL defines a collection of standard services are their interrelations as a basic starting point for further development by users. Functionalities like enhanced flight-plans, a mission control engine, data storage, communications management, etc. are offered. Additional services can be included according to requirements but all existing services and inter-service communication infrastructure can be exploited and tailored to specific needs. This approach reduces development times and risks, but at the same time gives the user higher levels of flexibility and permits the development of more ambitious applications.

1. Introduction

Current Unmanned Aerial Systems (UAS) technology offers feasible technical solutions for airframes, flight control, communications, and base stations. In addition, the evolution of technology is miniaturizing most sensors used in airborne applications. Hence, sensors like weather radars, SAR, multi spectrum line-scan devices, etc. in addition to visual and thermal cameras are being used as payload on board UAS. As a result UAS are slowly becoming efficient platforms that can be applied in scientific/commercial remote sensing applications (see Fig 1 for the most common subsystems in an UAS).

UAS may offer interesting benefits in terms of cost, flexibility, endurance, etc. Even remote sensing in dangerous situations due to extreme climatic conditions (wind, cold, heat) are now seen as possible because the human factor on board the airborne platform is no longer present. On the other side, the complexity of developing a full UAS tailored for remote sensing is currently limiting its practical application.

Even though the rapid evolution of UAS technology on airframes, autopilots, communications and payload, the generalized development of remote sensing applications is still limited by the absence of systems that support the development of the actual UAS sensing mission. Remote sensing engineers face the development of specific systems to control their desired flight-profile, sensor activation/configuration along the flight, data storage and eventually its transmission to the ground control. All this elements may delay and increase the risk and cost of the project. If realistic remote sensing applications should be developed, additional support to effective system support must be created to offer flexible and adaptable platforms for any application that is susceptible to

use them. In order to successfully accomplish this challenge, developers need to pay special attention to three different concepts: the flight-plan, the payload and the mission itself.

The actual flight-plan of the UAS should be easy to define and flexible enough to adapt to the necessities of the mission. The payload of the UAS should be selected and controlled adequately. The mission should manage the different parts of the UAS application with little human interaction but with large information feedback. Many research projects are focused on only one particular application, with specific flight patterns and fixed payload. These systems present several limitations for their extension to new missions.

This work introduces a flexible and reusable hardware/software architecture designed to facilitate the development of UAS-based remote sensing applications. Applications are developed following a service/subscription based software architecture. Each computation module may support multiple applications. Each application could create and subscribe to available services. Services could be discovered and consumed in a dynamic way like web services in the Internet domain. Applications could interchange information transparently from network topology, application implementation and actual data payload.

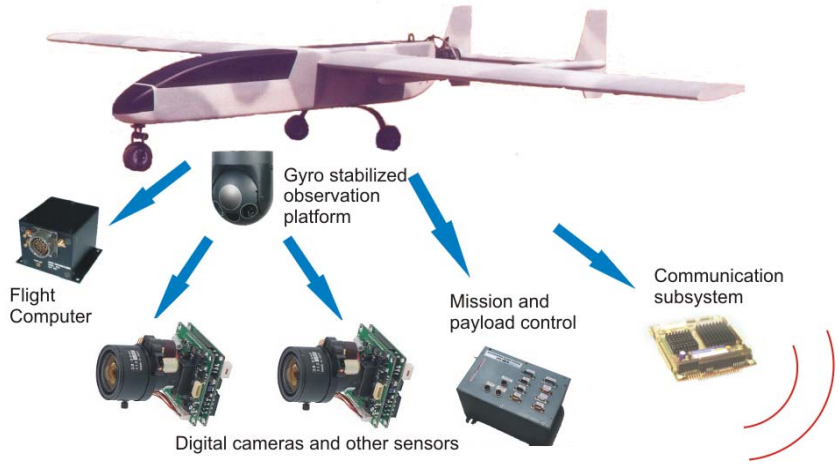

Figure 1: Common elements in a civil UAS.

This flexibility is organized into a user-parameterizable *UAS Service Abstraction Layer* (*USAL*). The USAL defines a collection of standard services are their interrelations as a basic starting point for further development by users. Functionalities like enhanced flight-plans, a mission control engine, data storage, communications management, etc. are offered. Additional services can be included according to requirements but all existing services and inter-service communication infrastructure can be exploited and tailored to specific needs. This approach reduces development times and risks, but at the same time gives the user higher levels of flexibility and permits the development of more ambitious applications.

This paper is organized as follows. Section 2 generally describes the underlying service oriented technologies that will be applied to create the USAL. Section 3 overviews the architecture of the USAL and describes the most relevant services that are included in the USAL to facilitate the development of UAS applications. Section 4 details the *Virtual Autopilot Service* (VAS) that permits the USAL architecture to abstract from autopilot details. Section 5 describes the *Flight Plan Manager* service that together with its RNAV based dynamic flight-plans constitute the core of the navigation capabilities inside the

USAL. Finally, Section 6 concludes the chapter and outlines future research and development directions.

2. System Overview

This section describes the architecture we propose for executing UAS civil missions: a distributed embedded system that will be on board the aircraft and that will operate as a payload/mission controller. Over the different distributed elements of the system we will deploy software components, called services, which will implement the required functionalities. These services cooperate for the accomplishment of the UAS mission. They rely on a middleware (Lopez et al. 2007) that manages and communicates the services. The communication primitives provided by the middleware promote a publish/subscribe model for sending and receiving data, announcing events and executing commands among services.

2.1 Distributed Embedded Architecture

The proposed system is built as a set of embedded microprocessors, connected by a Local Area Network (LAN), in a purely distributed and scalable architecture. This approach is a simple scheme which offers a number of benefits in our application domain that motivates its selection. Development simplicity is the main advantage of this architecture. Inspired in the Internet applications and protocols, the computational requirements can be organized as services that are offered to all possible clients connected to the network.

Extreme flexibility given by the high level of modularity of a LAN architecture. We are free to select the actual type of processor to be used in each LAN module. Different processors can be used according to functional requirements, and they can be scaled according to computational needs of the application. We denominate node to a LAN module with processing capabilities.

Node interconnection is an additional extra benefit in contrast with the complex interconnection schemes needed by end-to-end parallel buses. While buses have to be carefully allocated to fit with the space and weight limitations in a mini/micro UAS, the addition of new nodes can be hot plugged to the LAN with little effort. The system can use wake-on-LAN capabilities to switch on a node when required, at specific points of the mission development.

2.2 Middleware

Middleware-based software systems consist of a network of cooperating components, in our case the services, which implement the business logic of the application. The middleware is an integrating software layer that provides an execution environment and implements common functionalities and communication channels. On top of the middleware, services are executing. Any service can be a publisher, subscriber, or both simultaneously. This publish-subscribe model eliminates complex network programming for distributed applications. The middleware offers the localization of the other services and manages their discovery. The middleware also handles all the transfer chores: message addressing, data marshalling and demarshalling (needed for services executing on different hardware platforms), data delivery, flow control, message retransmissions, etc. The main functionalities of the middleware are: *service management*, *resource management*, *name management*, *communication management*.

Fig. 2 shows the UAS distributed architecture proposed. Services, like the Video Camera or the Storage Module, are independent components executing on a same node located on the aircraft. Also on board, there is another node where the Mission Control service executes. Both nodes are boards plugged to the LAN of the aircraft. The mission has also some services executing on ground. The figure also shows two of them: the Ground Station service and a redundant Storage Service.

Each node of the UAS distributed architecture executes also a copy of the Service Container software (see Fig. 2). The set of all Service Containers compose the middleware which provides the four functionalities described above to the application services. This includes acting as the communication bridge between the aircraft and the ground. The middleware monitors the different communication links and chooses the best link to send information to ground or to air. From the service views the middleware builds a global LAN network that connects the LAN on ground and the LAN on board.

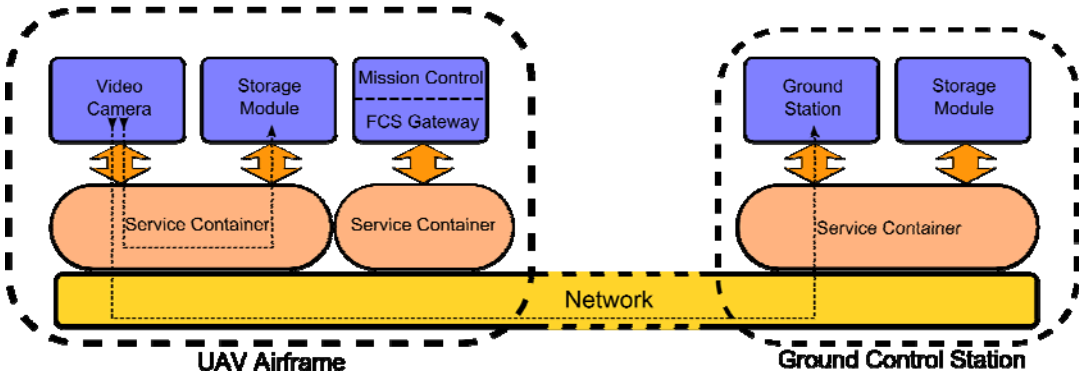

Figure 2: Overview of the architecture implementing the underlying middleware.

3. USAL: UAS Service Abstraction Layer

The existence of an open-architecture avionics package specifically designed for UAS may alleviate the developments costs by reducing them to a simple parameterization. The design of this open-architecture avionics system starts with the definition of its requirements. These requirements are defined by the type of UAS (mini or tactical UAS in our case) and the mission objectives. From the study and definition of several UAS missions, one can identify the most common requirements and functionalities that are present among them (UAVNET, 2005; RTCA, 2007; SC-203, 2007; Cox et. al. 2006).

These common requirements have been identified and organized leading to the definition of an abstraction layer called UAS Service Abstraction Layer (USAL). The goal of the USAL is twofold (Pastor et. al., 2007; Royo et. al., 2008): reduce "time to market" when creating a new UAS system; and simplify the development of all systems required to implement the actual mission.

3.1 USAL Service Types

Even though the USAL is composed of a large set of available services, not all of them have to be present in every UAS or in any mission. Only those services required for a given configuration/mission should be present and/or activated in the UAS. Available services have been classified in four categories according to the requirements that have been identified (see Fig. 3).

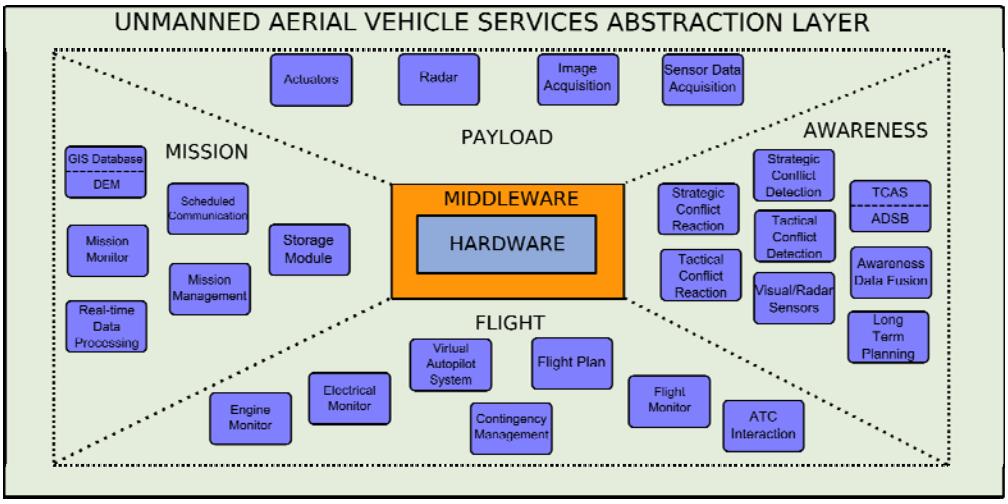

Figure 3: Overview of the USAL service-based architecture

The principal element is the computing system that manages the UAS flight. USAL considers the autopilot as a co-processor; it provides the system with a specific set of primitives that control the flight in the short term. The autopilot operation is supervised by a Flight Manager that abstracts users from autopilot peculiarities and offers flight plan specifications beyond classical way point navigation. Additional services help improving the security and reliability of the operation. The services in charge of the flying capabilities of the UAS are named *Flight Services*.

The next relevant elements are the components that should orchestrate the overall mission. This system may be joined by specific to mission additional systems like image processing hardware accelerators, etc. Storage and communication management should also be included by default. This set of standard plus user-defined services that control the mission intelligence are named *Mission Services*.

Successful integration of UAS in non-segregated aerospace will require a number of features to be included in the UAS architecture. Interaction with cooperative aircrafts through transponders, TCAS or ADS systems; and detection of non-cooperative aircrafts through visual sensors, should be implemented and the UAS must inform the pilot in command or automatically react following the operational flight rules for UAS that are currently being developed (EUROCONTROL, 2003; FAA, 2008). However, for certain cases, e.g. flying in segregated airspace, such services may not be necessary. Services that manage the interaction of the UAS with the surrounding airspace users, controllers or conditions are named *Awareness Services*.

Payload includes all those other systems carried on board the UAS. We divide them in data acquisition systems (or input devices) and actuators (or output devices). Input devices can be flight sensors (GPS, IMU, Anemometers) and earth/atmosphere observation sensors (visual, infra-red and radiometric cameras, chemical and temperature sensors, radars, etc.) Output devices are few or even do not exist in UAS civil missions because of the weight limitations: flares, parachutes or loom shuttles are examples of UAS actuators. Services controlling these devices are named *Payload Services*.

3.2 Flight Services

Many autopilot manufacturers are available in the commercial market for tactical UAS with a wide variety of selected sensors, sizes, control algorithms and operational capabilities. However, selecting the right autopilot to be integrated in a given UAS is a

complex task because none of them is mutually compatible. Moving from one autopilot to another may imply redesigning from scratch all the remaining avionics in the UAS. Moreover, the flight plan definition available in most autopilots is just a collection of waypoints statically defined or hand-manipulated by the UAS's operator. However, no possible interaction exists between the flight-plan and the actual mission and payload operated by the UAS.

Flight services are a set of USAL applications designed to properly link the selected UAS autopilot with the rest of the UAS avionics (Santamaria et al., 2008; Santamaria et al., 2007), namely the *Virtual Autopilot Service*, the *Flight Manager Service*, the *Contingency Service*, the *Flight Monitor Service*, etc. (see Fig. 4).

The *Virtual Autopilot Service* (VAS) is system that on one side interacts with the selected autopilot and is adapted to its peculiarities. VAS abstracts the implementation details from actual autopilot users. From the mission/payload subsystems point of view, VAS is a service provider that offers a number of standardized information flows independent of the actual autopilot being used.

The *Flight Plan Manager* (FPM) is a service designed to implement much richer flight-plan capabilities on top of the available autopilot capabilities. The FPM offers an almost unlimited number of waypoints, waypoint grouping, structured flight-plan phases with built-in emergency alternatives, mission oriented legs with a high semantic level like repetitions, parameterized scans, etc. These legs can be modified by other services by changing the configuration parameters without having to redesign the actual flight-plan; thus allowing the easy cooperation between the autopilot and the UAS mission.

The *Contingency Management* services are a set of services designed to monitor critical parameters of the operation (like battery live, fuel, flight time, system status, etc.). In case contingencies are detected, actions will be taken in order to preserve the security and integrity of the UAS: from flight termination, mission abort or system re-cycle.

3.3 Mission Services

The DO-304 RTCA (RTCA, 2007) describes 12 scenarios, selected from a list of 70 as representative of UAS missions. Scenarios are representative of different types of airframes and navigation conditions. Regarding their navigation procedures, we have classified them in three types, with increasing complexity: Static Area Surveillance, Target Discovering, and Dynamic Target Tracking.

In general, the most complex missions include the simplest missions as part of its objective. In example, in order to track a moving object, its previous discovering is needed, which is based on an area surveillance mission. For the Static Area Surveillance the only condition needed is the end-of-mission condition, but for the other two the end-users have to give the condition for Target Recognition.

The USAL offers a number of predefined services to implement a wide range of missions, namely the *Mission Manager*, the *Real-Time Data Processing*, the *Storage*, the *Scheduled Communications* the *GIS/DEM Database* and *Mission Monitor* (see Fig. 5). These services can be adapted to requirements by means of parameters (e.g. specific flight plan, sensors to be activated, information flows, etc), by adding specific software code to be executed or by adding specific user defined services. Next we describe in more detail some of them.

The *Mission Manager* (MMA) is the orchestra director of the USAL services. This service supervises the flight services and the payload services; as well as the coordination of the

overall operation. The MMA executes a user defined automata with attached actions (i.e. service activations) at each state or transition. Actions can be predefined built-in operations or specific pieces of user code. In particular the MMA is capable of modifying the actual flight plan by redefining its parameters or by defining new stages or legs.

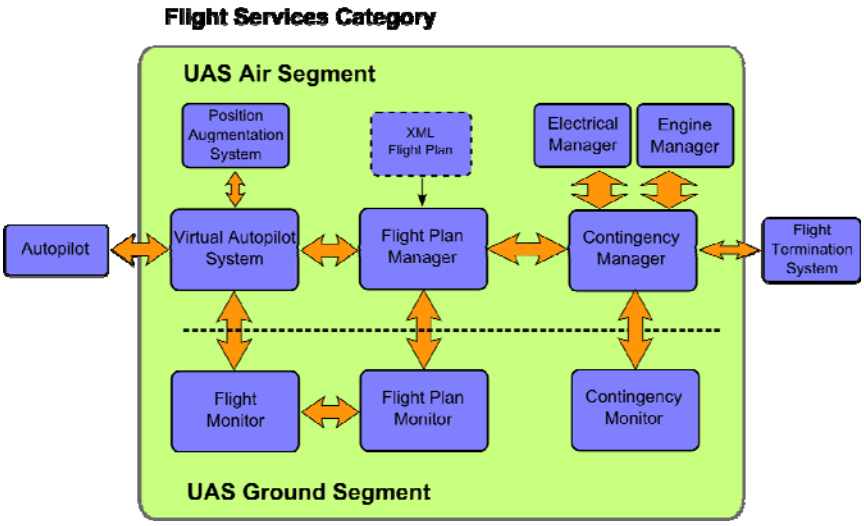

Figure 4: Overview of the available flight service category.

The *Real-Time Data Processing* (RDP) gives the intelligence for complex missions. The RDP offers predefined image processing operations (accelerated by FPGA hardware if available) that should allow the MMS to take dynamic decisions according to the actual acquired information.

The *Mission Monitor* (MMO) shows to end-users human friendly useful information about the mission. For example, during a wildland fire monitoring mission, it may present the current state of the fire front over a map. The MMO is basically executed on the ground and should be highly parameterized to fit the specific requirements of each mission.

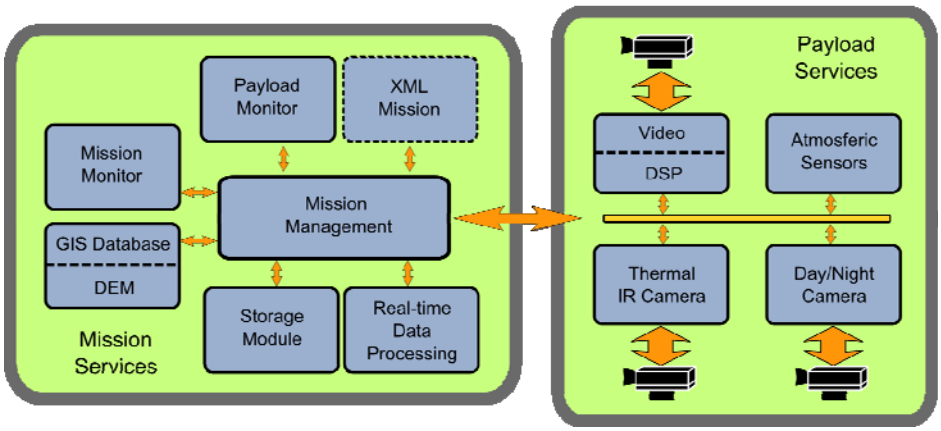

Figure 5: Overview of the available mission and payload service category.

3.4 Awareness Services

UAS must rely on its instrumentation equipment to properly inform the pilot in command on the ground or substitute the pilot capacities in VFR conditions. The awareness services are responsible of such functionalities. Flight Services are in charge of the aircraft management in normal conditions while the Awareness Services are in charge of monitoring surroundings conditions and overtake aircraft management in critical

conditions. In this case mission services come to a second priority, until flight conditions become again normal (see Fig. 6).

The *Awareness data fusion* (ADF) is a service designed to collect all available data about air vehicles surrounding our UAS, terrain and meteorological conditions. All this information can be obtained either by on board sensors or even through an external provider.

The *Tactical/Strategic Conflict Detection* service will analyze the fused information offered by the ADF in order to detect potential collision conflicts with objects/terrain/bad climate. Depending on the type of conflict, different types of reaction procedures will be activated. While reaction is executed it will keep monitoring than the conflict is really being avoided.

The *Tactical/Strategic Reaction* services, will implement avoidance procedures according to the severity of the conflict. Tactical reaction is designed in such a way it can overtake the Flight Plan Manager in order to execute a radical avoidance manoeuvre. Once completed, the FPM will regain control. A strategic reaction will command the FPM to slightly modify its selected flight plan trying to avoid the conflict but at the same time retaining the original mission requested by the Mission Manager.

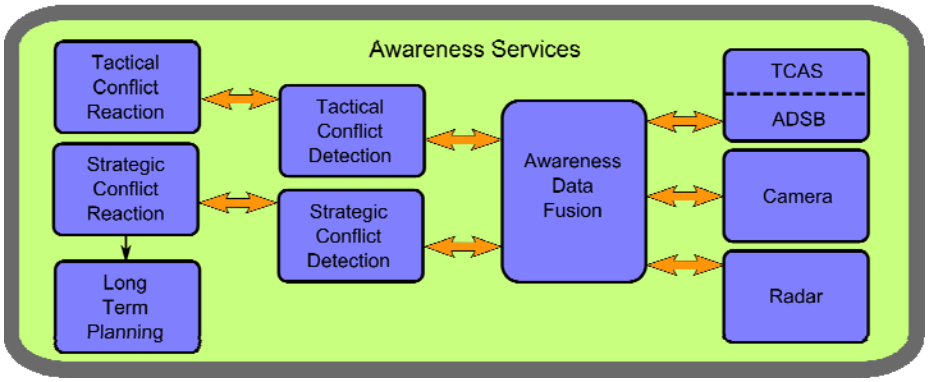

Figure 6: Overview of the available awareness service category.

3.5 Payload Services

Payload services are defined for low level devices, mainly raw data acquisition sensors that need to be processed before being used in real-time or stored for post-mission analysis. The complete list of services is directly related to available sensors, and except for most classical cameras they need to be created or adapted by the end user. However, USAL offers pre-build skeletons that should be easily adapted for most common devices.

4. The Virtual Autopilot Service (VAS)

The Virtual Autopilot Service is the component that will interact between the autopilot and the rest of the components of the USAL (Royo et. al., 2008). After studying several UAS autopilots we have seen that their functioning and capabilities are very similar, however their implementation details greatly differ (Haiyang et al., 2007). To improve the flexibility of a UAS is needed to abstract the concrete autopilot used. The main objective of the VAS is to implement this abstraction layer, to isolate the system of autopilot hardware changes. As every UAS mission needs to control autonomously the aircraft following a list of waypoints, the VAS clearly belongs to the minimum services required in the USAL.

4.1 VAS Architecture

VAS is a service that provides a standardized interface to the particular autopilot on board. Since it directly interacts with the selected autopilot it needs to be adapted to its peculiarities. VAS is a service provider that offers a number of information flows to be exploited. Given that not all autopilots are equal, VAS follows a contract between it as a service provider and its potential clients. VAS belongs to the set of services defined in the USAL and will provide flight monitoring and control capabilities to other services.

The inclusion of the VAS greatly improves the flexibility of the system. The autopilot unit can be replaced by a new version or a different product, but this change will have no impact on the system except for the VAS. Another important motivation is to provide an increased level of functionality. VAS should permit operation with a virtually infinite number of waypoints, thus overcoming a limitation present in all studied UAS autopilots.

Flight Monitoring Services

Autopilot manufacturers group all this information in large packets of data, which are sent via a radio modem at a certain frequency. In our service-based architecture, the VAS service will offer this information over a LAN to all the services that need this information. The information will be semantically grouped in a way that this information relates to parameters, situations or attitudes of the aircraft, independently of the real autopilot hardware and sensors.

Navigation Services

The FPM is in charge of generating the navigation commands to the VAS. In most cases these commands will take the form of waypoints or requests for changing the autopilot state. The VAS feeds the autopilot with its internals waypoints as it consumes system waypoints and commands. This information will be used by the FPM in order to interact dynamically with VAS and the mission services. In case the FPM or some other service implied in the UAS flight fails, the VAS can take control of the UAS. The VAS also offers configuration parameters for the autopilot operation, as well as parameters to configure the operative of the states in which the VAS may function.

VAS and Autopilot Status/Alarms

The VAS is the interface between the autopilot and the rest of the system. So is in charge of inform the status of the autopilot and its own status. An autopilot is a complex hardware that needs to be monitoring every time. With this group of packets we can monitor the autopilot and the VAS status; when any part of these devices has a failure the VAS will send an alarm to the network. All of these alarms are sending by the network as events for two reasons. First, because the alarms are very important for the system and it is needed that these notifications safely arrive to all the services that process them. Second, we will only need to know the status when something is wrong.

4.2 VAS Operational States

In this section we are going to describe a general overview of the VAS operational states. For lack of space, the detailed the description of the states and the transitions are omitted. The commercial autopilots are much focused in flight states, however a mission can be composed of many different states, for example, and we can have different behaviours for the contingencies, which need different types of response. Many autopilots solve this sort of problems just coming back to base station. However, we want the UAS to be able to enter in safe states where it can try to recover the situation.

Fig. 7 shows all the VAS states. In each state, the UAS develops several tasks in order to achieve the goals of the mission in each moment. All the related states are grouped together. The initial state inside each group is showed with an arrow on the top right box state corner. The rest of the arrows show the transitions between the different states.

Start-Up States. The first states related to system initialization are arranged in the Start-Up group. The initial state when the system is switched on is the Stop state. In this state, the UAS is stopped with all the devices of the system on. The next state is the Configure and Check state in which UAS begins to configure itself. When all services are properly configured, the UAS can change to following state: parking. Some sensors, servo mechanisms and communication modules will be checked again when the engine is working in addition to engine parameters, then the UAS can go on to taxi states.

Taxi States. At this moment, the UAS is working and needs to proceed to the runway to start the mission. The VAS can make this operation in two ways, as auto taxi and manual taxi. While the UAS is on ground the UAS speed will be limited. Also, if the UAS is in auto taxi and anything is wrong the VAS can give the control to the operator changing to manual taxi. When the UAS is in the runway heading then the VAS can pass to the take-off states.

Take-off States. After the taxi states the UAS is prepared to start the mission. The mission starts with the take off state; this operation is one of the most dangerous because the aircraft begins to fly and the UAS usually does not have enough altitude to response of any contingency. This operation can also be developed manually or automatically.

Navigation States. At this point of the diagram, the UAS is flying to a secure altitude. When the UAS has arrived to this altitude, it will change automatically to Waypoint Navigation which it is first state of the navigation group. The navigation states are composed by waypoint navigation state, directed state, hold at state and manual state.

In waypoint navigation the UAS follows the waypoints that have been uploaded previously in the VAS. While the UAS is in directed state it will maintain a specific altitude, airspeed and bearing. When the UAS is in Hold At state, it executes a hold pattern around the indicated waypoint. Finally, in manual state the operator has direct control over the airframe's control surfaces. The operator can switch from one Navigation state to each other as he commands from the ground station.

Landing States. When the mission has successfully finished, the UAS has to back home and prepare landing. In order to carry out this task the UAS switches to landing states. This group is composed of land pattern state, land abort state, auto land and manual land. During the land pattern state, the UAS will fly an approximation pattern in order to prepare for the landing. Then, we can choose between auto landing and manual landing.

Safe States. If any failure occurs during the navigation states, the VAS can switch to the safe states. These states are composed by safe hold state, safe return state and safe land state. When we have a failure in the system, the UAS will change the state to the Safe Hold. In this state the UAS will remain trying to recover the failure. After a timeout the UAS will switch to Safe Return state. In this state the UAS will return to the closest emergency runway in order to start the pattern landing.

Reaction State. The Safe Reaction is on charge of analyzing the environment around the aircraft and generating reactions in case of a quick evasive response is needed. When the Awareness services detect a dangerous situation, it generates an alarm and the VAS switch

to Safe Reaction. In this state the UAS will be commanded by the Tactical Reaction service to solve the problem.

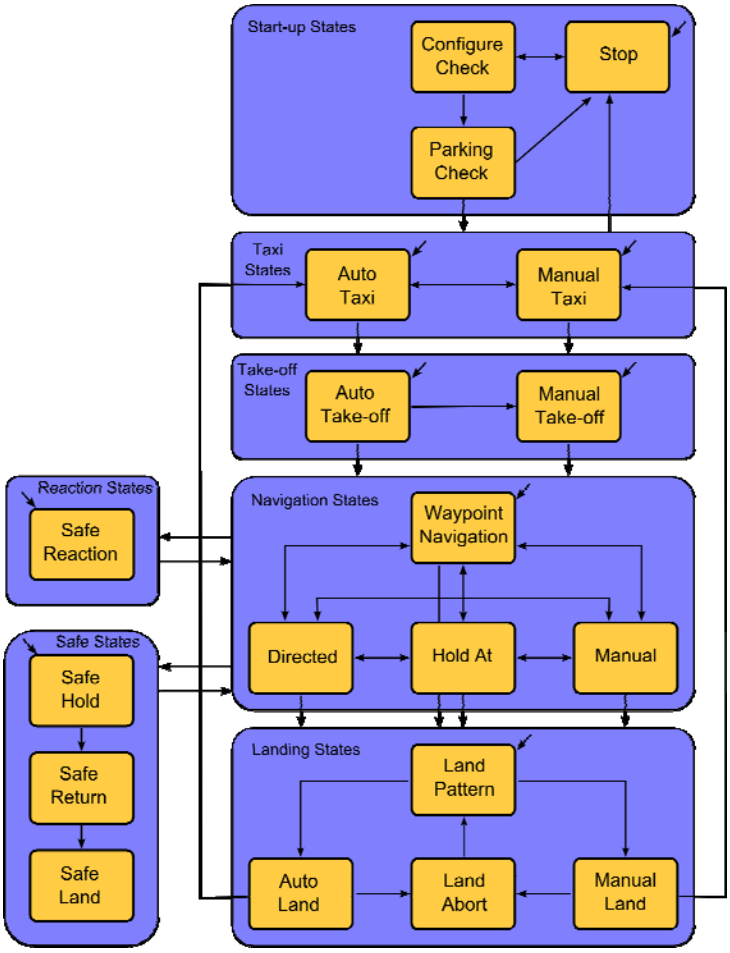

Figure 7: VAS Operational States and Transitions.

5. Mission Aware Flight Planning

Previous sections have introduced the proposed UAS architecture. In this section we detail the specification mechanism that will be used to describe the UAS flight plans. Most current UAS autopilot systems rely on lists of waypoints as the mechanism for flight plan specification and execution. This approach has several important limitations: (1) It is difficult to specify complex trajectories and it does not support constructs such as forks or iterations. (2) It is not flexible because small changes may imply having to deal with a considerable amount of waypoints and (3) it is unable to adapt to mission circumstances. Besides (4) it lacks constructs for grouping and reusing flight plan fragments. In short, current autopilots specialize in low level flight control and navigation is limited to very basic go to waypoint commands.

5.1 Flight Plan Overview

Area navigation (RNAV) is a method of navigation that takes advantage of the increasing amount of navigation aids (including satellite navigation) and permits aircraft operation on any desired flight path. RNAV procedures are composed of a series of smaller parts called legs. To translate RNAV procedures into a code suitable for navigation systems the industry has developed the "Path and Termination" concept. Path Terminator codes

should be used to define each leg of an RNAV procedure. Leg types are identified by a two letter code that describes the path (e.g., heading, course, track, etc.) and the termination point (e.g., the path terminates at an altitude, distance, fix, etc.).The flight plan represents the instructions that will be given to the FP Manager. A flight plan follows a hierarchical structure and is composed of stages, legs and waypoints.

Stages are the largest building blocks within a flight plan. They organize legs into different phases that will be performed in sequence. Legs specify the path that the plane must follow in order to reach a destination waypoint from the previous one. A *waypoint* is a geographical position defined in terms of latitude/longitude coordinates. Waypoints can be named or unnamed depending on whether they are associated to a fix. A fix corresponds to a geographical position of interest with a name and description.

Stages correspond to flight phases that will be sequentially executed. Fig. 8 shows a sequence of a flight plan with all stages in the context of a fire fighting mission. The flight plan starts with a Take-Off. Once the aircraft is airborne it will perform a Departure Procedure that will connect with an En Route leading to the forest fire site. Upon arrival the Mission stage will start. When this stage concludes the UAS will enter another En Route stage leading to the landing area. There an Arrival Procedure will be executed to connect with the final Approach and Land stages. Each one of these stages will be composed of a set of legs as described in the following paragraphs.

5.2 Detailed Leg description

A *leg* specifies the flight path to get to a given waypoint. In general, legs contain a destination waypoint and a reference to their next. Most times legs will be flown in a single direction, but within iterative legs reverse traversal is also supported. There are four different kinds of legs.

Basic Legs. A number of basic legs are available to the flight plan designer. They are referred to as basic legs to differentiate them from control structures like iterative or intersection legs and parametric legs. All of them are based on already existing ones in RNAV: *Initial Fix*, *Track to a Fix*, *Direct to a Fix*, *Radius to a Fix* and *Holding Pattern*.

Iterative Legs. A complex trajectory may involve iteration, thus the inclusion of iterative legs. An iterative leg has a single entry (i.e. its body can be entered from a single leg), a single exit and includes a list with the legs that form its body. Every time the final leg is executed an iteration counter will be incremented. When a given count is reached or a specified condition no longer holds the leg will be abandoned proceeding to the next one.

Intersection Legs. Intersection legs indicate points where two or more different paths meet and where decisions on what to do next can be made. All joins and forks will end and start at an intersection leg.

Parametric Legs. Dynamic characteristics of the mission environment will make previous leg types insufficient. Parametric legs provide an increased level of adaption to changes that occur during mission time. With parametric legs the flight path is dynamically generated according to input values. Parametric legs offers: complex trajectories that can be generated with no need to specify a possibly quite long list of legs; and the UAS path can dynamically adapt according to the input values.

Conditions. There are several points in the flight plan where conditions can be found: namely in holding patterns, iterative and intersection legs. For intersection legs, they are necessary in order to determine what path to follow next. For the rest of legs they will let

the FPM know when to leave the current leg and proceed to the next one. Conditions will not be directly specified in the flight plan. Instead, each leg that depends on a condition will contain a reference which will be used to identify the condition. Conditions will be stored and processed separately. When the outcome of a condition is set the FPM will be notified so that this change is taken into account for waypoint generation.

5.3 Flight Plan Manager

This section presents the USAL service responsible for processing and executing the proposed flight plans: the Flight Plan Manager Service (FPM). The FPM forms part of a wider collection of services that provide the UAS with all its capabilities. The FPM belongs to the flight services category, it will collaborate with other on-board and remote services in order to execute the given flight plan.

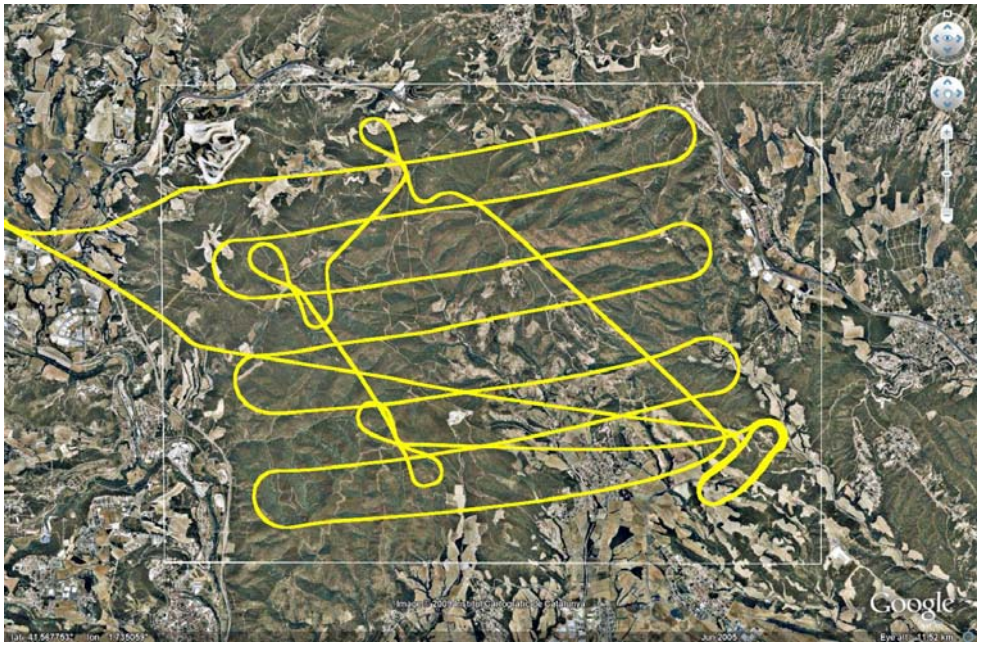

Figure 8: Flight Plan Application Example.

In order to execute the flight plan the FPM will send navigation commands to the VAS. These commands mainly consist in waypoints the aircraft has to fly to. Since the flight plan is specified in terms of legs some translation process is needed for converting them into the waypoint sequences expected by the VAS.

Computing waypoint sequences that approximate the legs specified in the flight plan is the main task of the FPM execution engine. This flow of waypoint commands is the main form of interaction between the FPM and the VAS. Sometimes, these waypoints will be accompanied by additional fields indicating speed and altitude change requests.

One of the main features of our flight plan specification mechanism is that it enables the UAS to adapt to mission circumstances. This can be done in two manners: first, with the possibility of using conditions in different leg types. Second, by using parametric legs, whose final form depends on the values of the input parameters.

Conditions mark a point where a decision can be made about what path to follow next. The decision-making process is not directly done by the FPM. The flight plan contains references to condition identifiers. The outcome of these conditions will be set by the Mission Control Service. The MCS will also decide what the actual parameter values for

parametric legs are. These functions are currently done from the GCS but the inclusion of the MCS will provide the UAS with a high degree of automation.

Waypoint generation works in a decoupled manner from the FPM operation depicted in Fig. 9. The FPM is implemented following a producer-consumer model. Each new waypoint is stored in a queue. The consumer will pass the waypoints from the queue on to the VAS. The head of the queue contains the waypoint the VAS is heading to. The queue is also used to keep track of unsent waypoints.

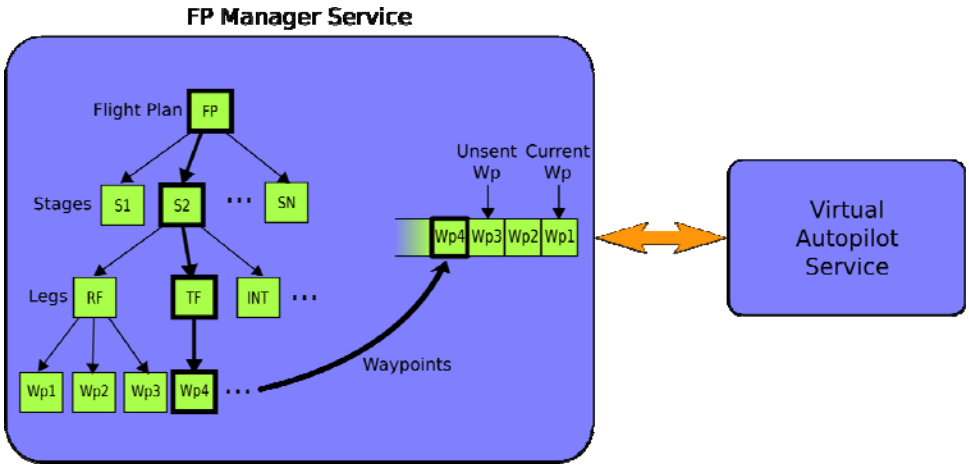

Figure 9: Overview of the Flight Plan execution procedure by the FPM.

6. Conclusions

USAL is a service-oriented architecture designed to support the development of remote sensing applications. USAL offers a number of pre-defined services that can be easily parameterized to the specific needs of the application. Additional services can be introduced to incorporate new functionalities and at the same time reusing available services. A middleware designed to support the type of inter-service communications required by the USAL is also introduced.

A novel flight plan specification to be employed within the USAL has been also introduced. The proposed specification mechanism improves on the common list of waypoints approach by providing RNAV-like legs as the main unit for flight plan construction. Besides the leg concept is extended to include higher level control structures for specifying iterative behaviour and branching. The decision making for this control structures can be based on mission variables, therefore enabling the UAS to respond depending on mission development.

7. Acknowledgments

This work has been partially funded by Ministry of Science and Education of Spain under contract CICYT TIN 2007-63927 and by the European Organisation for the Safety of Air Navigation (EUROCONTROL) under its Research Grant scheme. The content of the work does not necessarily reflect the official position of EUROCONTROL on the matter.

8. References

W3C Working Group (2004), W3C Note 11: Web Services Architecture, *http://www.w3c.org/TR/ws-arch*, February 2004

UPnP Forum (2008), UPnP Device Architecture 1.0, http://www.upnp.org/specs/arch, April 2008

UAVNET Thematic Network (2005), European Civil Unmanned Air Vehicle Roadmap http://www.uavnet.com, March 2005

RTCA (2007), DO-304: Guidance Material and Considerations for Unmanned Aircraft Systems, March 2007

Santamaria, E.; Royo, P.; Lopez, J.; Barrado, C.; Pastor, E. & Prats, X. (2007). Increasing UAV capabilities through autopilot and flight plan abstraction, *Proceedings of the 26th Digital Avionics Systems Conference*, Dallas (TX), October 2007, AIAA/IEEE

Santamaria, E; Royo, P.; Barrado, C.; Pastor, E.; Lopez, J. & Prats, X. (2008). Mission Aware Flight Planning for Unmanned Aerial Systems, *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu (HI), August 2008, AIAA

Pastor, E.; Lopez, J. & Royo, P. (2007). UAV Payload and Mission Control Hardware/Software Architecture. *Aerospace and Electronic Systems Magazine*, Vol.22, No.6, June 2007 3-8

Lopez, J.; Royo, P.; Pastor, E.; Barrado, C. & Santamaria, E. (2007). A Middleware Architecture for Unmanned Aircraft Avionics, *Proceedings of 8th Int. Middleware Conference*, NewPort (CA), November 2007, ACM/IFIP/USEUNIX

Royo, P.; Lopez, J.; Pastor, E.; & Barrado, C. (2008). Service Abstraction Layer for UAV Flexible Application Development, *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 2008, AIAA

Cox, T.H.; Somers, I. & Fratello D.J. (2006). Earth Observations and the Role of UAVs: A Capabilities Assessment, 2006, NASA

Haiyang, C.; Yongcan, C. & YangQuan, C. (2007). Autopilots for Small Fixed-Wing Unmanned Air Vehicles: A Survey," *Proceedings of International Conference on Mechatronics and Automation (ICMA)*, Harbin, China, 2007, IEEE

FAA (2008). Aeronautical Information Manual, Official Guide to Basic Flight Information and ATC Procedures, *U.S. Federal Aviation Administration*, 2007.

EUROCONTROL (2003). Guidance Material for the Design of Terminal Procedures for Area Navigation, *European Organisation for the Safety of Air Navigation*

Schmidt, D.C. (2002). Middleware for Real-Time and Embedded Systems, *Communications of the ACM*, June 2002