SERVICE-ORIENTED ARCHITECTURE TO INTEGRATE FLIGHT SAFETY AND MISSION MANAGEMENT SUBSYSTEMS INTO UAVS

Veronica Vannini[a] , Jesimar da Silva Arantes[a] , André Pierre Mattei[b] , Nina Figueira[c] , Márcio da Silva Arantes[b] , Claudio Fabiano Motta Toledo[a] , Onofre Trindade Júnior[a] , Pierre de Saqui-Sannes[d]

[a]University of São Paulo, Av. Trabalhador São Carlense, 400 - São Carlos - SP - Brazil , [b]Senai Innovation Institute for Embedded Systems, Av. Luiz Boiteux Piazza, 1302 - Florianópolis - SC - Brazil , [c]Brazilian Army, Dept. of Science and Technology, Quartel-General do Exército - Bloco G - 3o Andar - Brasília - DF - Brazil , [d]Institut Supérieur de l'Aéronautique et de l'Espace, 10 Avenue Edouard Belin, 31400 - Toulouse - France

Keywords: *Unmanned Aerial Vehicle, Service-Oriented Architecture, Autonomous Flight, Embedded System*

Abstract

This paper presents the design and implementation of a Service-Oriented Architecture (SOA). This architecture tackle with the data exchange different elements onboard the Unmanned Aerial Vehicle (UAV) and the real-time safety-critical operation of In-Flight Awareness Augmentation System (IFA^{2}S) and non-safety critical Mission Oriented Sensors Array (MOSA), respectively to take care of flight safety and mission accomplishment. The IFA and MOSA systems implemented and evaluated using the Software-In-The-Loop (SITL) simulation technique. The case study conducted had several critical situations were assessed through the sensors, then a diagnosis was made, and finally, a decision was made as a safety measure.

1 Introduction

The recent advances in onboard systems of Unmanned Aerial Aircraft (UAV) aims both flight safety and mission management, where In-Flight Awareness Augmentation System (IFA^{2}S) [1] and Mission Oriented Sensors Array (MOSA) [2] are examples of systems focused on these aspects. In order to be integrated into an aircraft, these subsystems shall comply with many different requirements such as time constraints, share computational resources and share the data from several sensors [3]. In order to enable flexibility, improve data management efficiency and become easier future integration (either services or hardware), it is necessary to have an environment capable of dealing with different tasks, protocols, time constraints, and priorities. Service-Oriented Architecture (SOA) provides this environment using a middleware that enables information interchange between services on a loose coupling [7] and [8].

Although IFA^{2}S and MOSA have different purposes (flight safety improvement and mission accomplishment efficiency, respectively), most data used by both come from same sensors and they share most of the information. Both systems act as service providers and consume data from sensors, which are also considered as service providers. The subsystem IFA^{2}S is a safety critical, real-time system designed to semi-automatically (with human supervision) identify and avoid flight hazards and accidents. These problems come from either internal or external

causal factors. MOSA is a non-safety critical system designed to manage the mission accomplishment by decoupling the mission-oriented part of the system from the aircraft control systems (safety-critical).

The IFA^{2}S is a novel autonomous decision-maker onboard the aircraft aiming to improve flight safety. It makes the UAV more conscious (situational awareness increase) about its subsystems conditions (internal health), flight profile, intruders presence (other aircrafts), and surrounding conditions (ground and meteorological) by keeping pilots on the ground as system managers. It provides a platform that is Situational Awareness (SA), instead of relying on human pilots' perceptions. It allows the system to act as soon as it identifies a situation that potentially leads to an accident.

MOSA is an architecture created to improve mission management during the flight [2]. The integration of MOSA with a set of sensors provides information for specific applications. MOSA also manages mission accomplishment during flight and interfaces with the safety-critical part of the UAV (automatic pilot). In addition to the hardware, a MOSA system also includes the software necessary to carry out a mission, communicate with all sensors, and send/receive data to the aircraft.

The MOSA and IFA^{2}S systems have been designed in a modular way allowing that new planning and replanning algorithms may be added to the system without significant modifications [4]. Another feature of the platform is to be plug-in-play by allowing new sensors and hardware equipment to be easily coupled by changing only a few configuration files and a few system changes in some cases. Another advantage of the platform is the facility of making changes in the missions without a high cost of time. Although the MOSA and IFA^{2}S systems are general and can be used in several UAVs, they have been thought to run at low cost and low weight UAVs.

The present paper is organized as follows: Section 2 presents some related work and Section 3 defines the problem to be studied. The proposed SOA architecture is introduced in Section 4 and experimental results are reported in Section 5. The conclusion of this work follows in Section 6.

2 Related Work

A Service Oriented Architecture (SOA) was presented in [9] for UAVs with the objective of automating the control of mosquito that are vectors of diseases such as malaria and dengue. A system of surveillance, control, suppression and elimination of vectors is introduced. The authors propose a software solution, where the Auto Pilot (MedizDroids) is specialized for the problem approached. Unlike the work developed here, the work in [9] does not present an automatic route generation system for external environments, nor a safety system, nor it distinguishes the specific system from the mission (control) The control system (AP) becomes difficult to use the aircraft on other missions.

The work on [10] has developed a cloud system for the control of one or multiple drones applied to surveillance missions by tracking the GPS position of the device to be followed. The proposed architecture is mainly based on Internet-of-Drones (IoD), where the application in the ground only deals with Web services, the interface with the cloud and the human-aircraft interface. The application in the aircraft is reserved for the control of the web services, the interface with the cloud and hardware. All processing, mission control, data collection and UAV monitoring is allocated to the cloud application. Despite some similarities with the project proposed in this article, the proposal of [10] has no route planning for obstacle avoidance as it does not have an aircraft health surveillance system.

A mission-oriented architecture is presented in [11], which proposes a modular and generic system for the control of UAVs using computational vision in agriculture. The authors present a system that changes the current mission objective and implements corrections in the flight path after finding a new target. In this work, the companion computer Odroid U3+ was integrated with the AP Pixhawk and a Micro Arduino to control a quad-

copter. A mission-oriented system is also presented in [11], but no flight safety system or aircraft analysis or route planning is proposed.

3 Problem Description

One of the proposals of this work is to develop an autonomous UAV by incorporating in it an onboard computer, so it can autonomously perform missions. All mission management, safety and route is done on board the own aircraft.

Several sensors/systems placed aboard the aircraft permit the system to be capable of detecting flight-critical failures such as checking low battery level, GPS loss, AutoPilot (AP) failures. Thus, some decision-making needs to be made for aborting the current route and executing, for example, an emergency landing over a safe region, a vertical landing, a Return To Launch (RTL), or open the parachute as a last resort.

Although the platform operates autonomously, the mission designer should describe what he or she wants the aircraft to do such as informing the launch site, the mission's objective waypoints, the landing place, and where obstacles are in the scenario. All decision-making, for example, can decide how to achieve the objectives of the mission by making the deviation of obstacles. Thus, the MOSA system is able to do this autonomously. The IFA^2 system can ensure that the aircraft during the flight executes a fast emergency decision under any critical failure.

Figure 1 shows an example of the type of problem studied and how the MOSA and IFA^{2}S systems solve this problem autonomously. The mission plan to be executed and the tasks of the MOSA and IFA^{2}S systems are detailed next.

Mission Plan: The UAV should start its flight at the point P_1 (*takeoff*) and proceed to the goal points P_2, P_3, ..., P_6 and conclude it at P_7 (*landing*). This sequence of points determines the whole mission to be followed by the UAV that will take pictures at each region. The scenario has two obstacles and No-Fly Zones (NFZ), which give the boundaries of the aircraft's navigable environment.

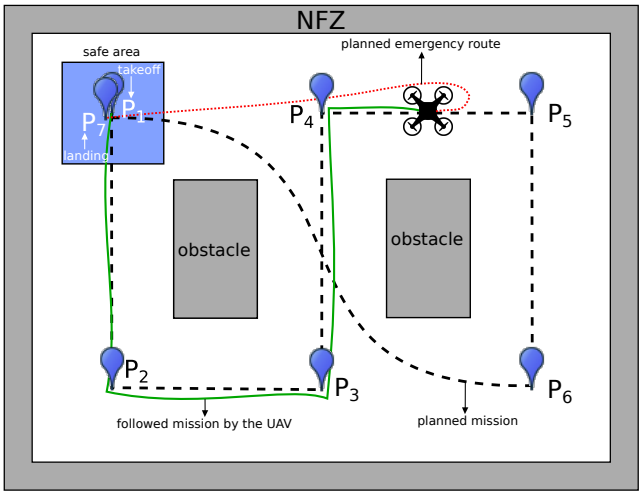

Fig. 1 Illustrative scenario of the mission plan managed by IFA^{2}S and MOSA systems.

MOSA System: MOSA first plans the route that minimizes the aircraft's fuel consumption between waypoints P_1 and P_2, and between waypoints P_2 and P_3 and so on until P_6 and P_7. Next, the system oversees the execution of the route by autopilot (green line in Figure 1) as well as the right instant to start taking pictures over each waypoint.

IFA^{2}S System: The UAV may present a failure such as low battery. The supervisor system (IFA^{2}S) detects the failure and decides to abort the current mission. Therefore, the UAV control executed by MOSA is halted and an algorithm for path re-planning is executed. The new trajectory is now overseen by IFA^{2}S (red line in Figure 1) landing the aircraft over a safe region (blue region).

Based on the mission specified in Figure 1, we can define a set of actions that the MOSA and IFA^{2}S systems must execute to accomplish the mission. Figure 2 synthesizes through a flowchart the set of actions in which MOSA performs the calibration of the systems and performs the path planning between points P_1 to P_7. Next, the aircraft takeoff and rises until reaching cruising altitude, follows the waypoints of the mission (P_1 to P_7) and finally lands (P_7). In parallel with MOSA, the IFA^{2}S system also performs system calibration (for example, read AP parameters and get home location), and monitors the sensors for

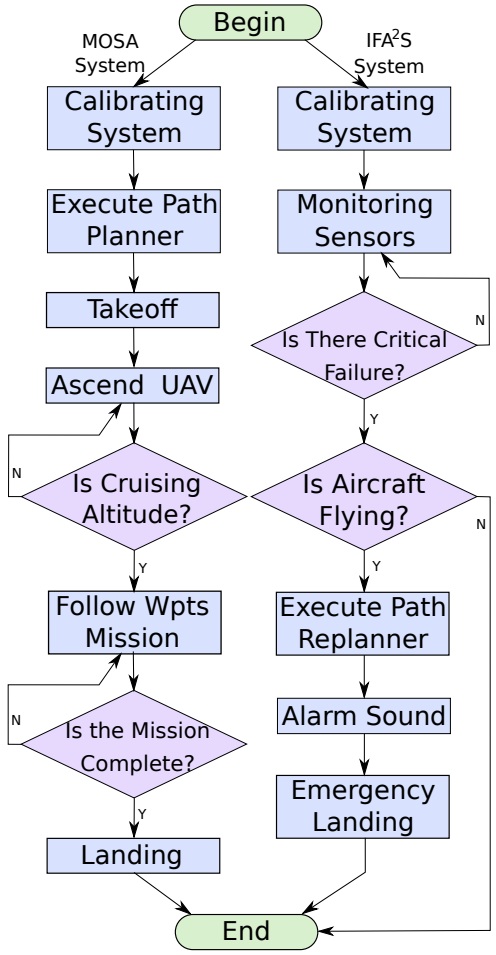

Fig. 2 Execution flowchart of the MOSA and IFA^{2}S systems.

critical situations. If the aircraft is flying in the event of any failure, a new route is calculated, the sound of an alarm goes off, and the aircraft lands on that region aborting the current mission.

4 Service-Oriented Architecture

The design of the Real-Time Service-Oriented Architecture was accomplish by using Model Based Space System Engineering (MBSSE) with SysML as language for system definition [13], and the free software TTool [12]. The MBSSE is chosen for this project since it facilitates application of concurrent engineering in the early phases of the aerospace system life cycle, Phases 0, A, and B [14]. Phase 0 refers to mission analysis and needs identification. Phase A is a feasibility study containing possible system concepts and assess its technical and programmatic aspects. Phase B establishes a preliminary design definition by confirming the technical solutions using trade-off studies for the selected system concept.

The SOA is responsible for the integration of service providers (e.g. GPS, cameras, accelerometers, gyros and barometers) and high-level systems such as IFA^{2}S, MOSA, the autopilot service providers and data consumers. It is implemented by using a deterministic design (same input always leads to the same output) capable of respecting time constraints in order to carry out common features and data exchange.

The development of SOA by using a middleware can be seen on the architecture depicted by the SysML block diagram in Figure 3. This software architecture provides the autonomy necessary to make the platform more capable for both managing its mission (non safety-critical service) and avoiding dangerous situations (safety-critical service). The design controls services and provides are both resources and priorities necessary for task accomplishment.

The SOA middleware implementation explores similarities between services and makes it easier information access. The Resource Manager is the integration interface for sensors, IFA^{2}S, and MOSA and uses standard interface SSP/SSI (Smart Sensor Protocol/Smart Sensor Interface). MOSA, IFA^{2}S, and Reroute Planner use sensor's data for different purposes and the Resource Manager makes them available. Reroute Planner updates the mission route due to either an emergency or mission update. The Admission Controller supervises the access to both Reroute Planner and Resource Manager giving priority to IFA^{2}S in case of conflict with MOSA. IFA^{2}S is an event based service and has priority over MOSA when either sending orders or requesting data. The direct connection between IFA^{2}S and the flight control surfaces aims prioritize emergency landing in the case of autopilot failure.

The communication between the IFA^{2}S and MOSA systems occurs through the TCP/IP network protocol using sockets. The IFA^{2}S sys-

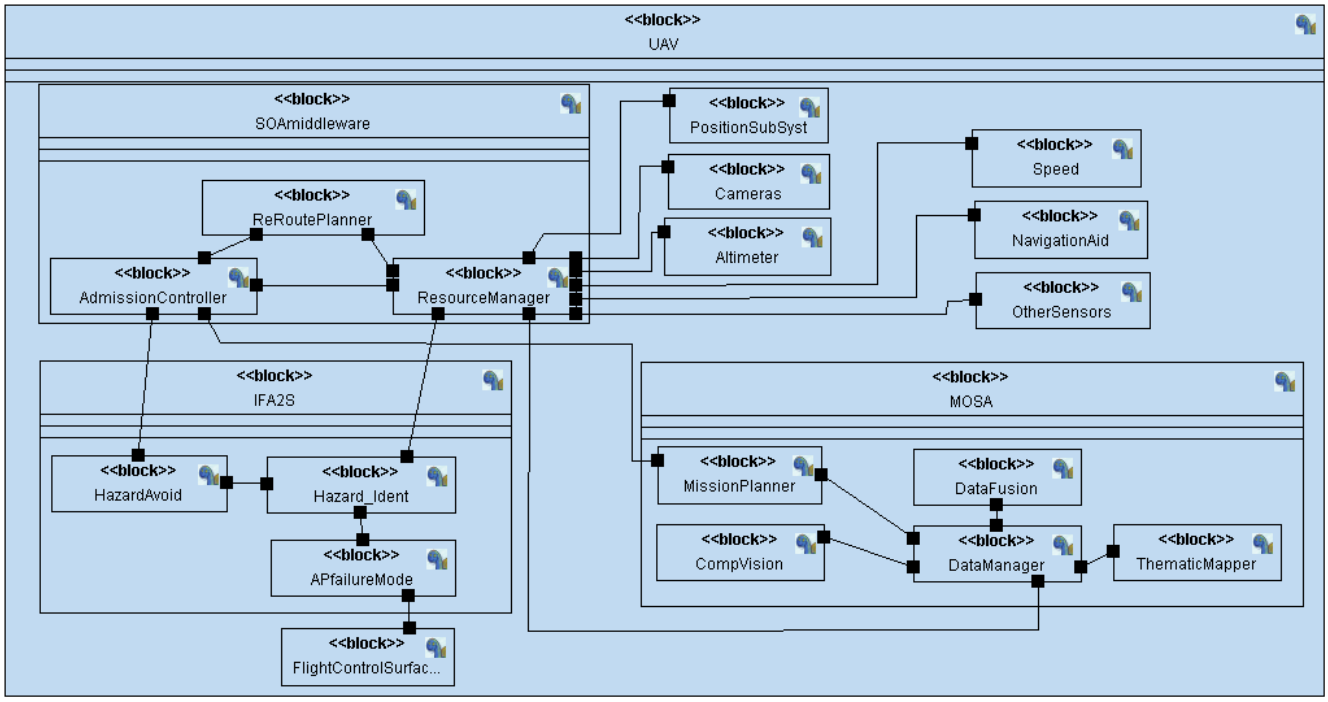

Fig. 3 SOA architecture in which the MOSA and IFA^{2}S systems were integrated into the UAV.

tem is the server and MOSA is the client in the application, indicating the order of initialization of each system. Figure 4 shows in detail how the messages are sent. Once the MOSA system started, it will inform the IFA^{2}S. Next, IFA^{2}S informs the home location of the aircraft (important parameter during the flight). Subsequently, the IFA^{2}S informs that MOSA can start the calculation of the route to follow. As soon as this calculation finishes, MOSA tells IFA^{2}S that the route execution has already started. During the mission, if the IFA^{2}S detects a fault, the MOSA is stopped and an emergency route is calculated by IFA^{2}S. Finally, if during the execution of the mission any failure occurs in MOSA, the IFA^{2}S is warned about such occurrence and initiates an emergency route.

A robust detection, diagnostic, and decision-making system were implemented on the IFA^{2}S platform. To detect some event of abnormal behavior in the system, specifics sensors or a software system must be used. A diagnostic step is initiated based on the sensing and system data. Lastly, basis in the diagnosis, an action must be done by the decision-making system. Figure 5 shows the steps involved from the sensing to the decision-making by the IFA^{2}S system. It is important to note that the platform described has not implemented some of the strategies that are to be made as future work. In the following example, based on Figure 5, it is assumed that the aircraft has a Power Module system capable of monitoring the system's current battery level. Through this sensor, low battery diagnosis becomes simple by comparing the current battery level with a threshold. In case the value obtained is smaller than the threshold, the system must take a security action. The possible actions taken by the system are to make the emergency landing, by firing an audible alarm alerting people on the ground that there has been a critical failure and that aircraft will land quickly.

In Figure 5, other sensors/systems were embedded on the platform such as GPS, Autopilot, MOSA and IFA^{2}S and Weather Forecast. Through these components, different diagnostics can also be evaluated. For instance, the location system may not have 3D fix, showing a GPS failure, and a vertical landing should be started immediately. A bad weather condition may reveal

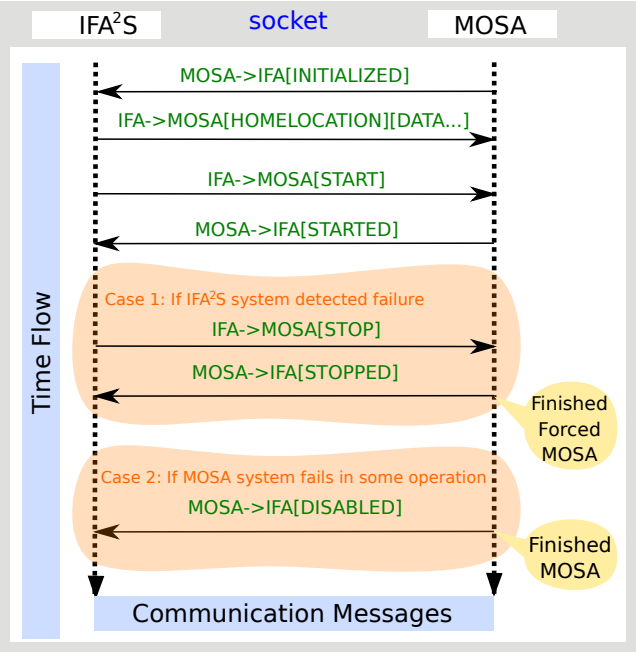

Fig. 4 Communication system between MOSA and IFA²S.

that the aircraft should abort the current mission and return to base (RTL). A set of different sensing, diagnostics, and actions have been evaluated and are described in the results section.

5 Experimental Results

A set of simulated experiments were conducted to evaluate our architecture. They were calibrated based on a quadcopter and the results are reported below. The scenario evaluated has dimensions of

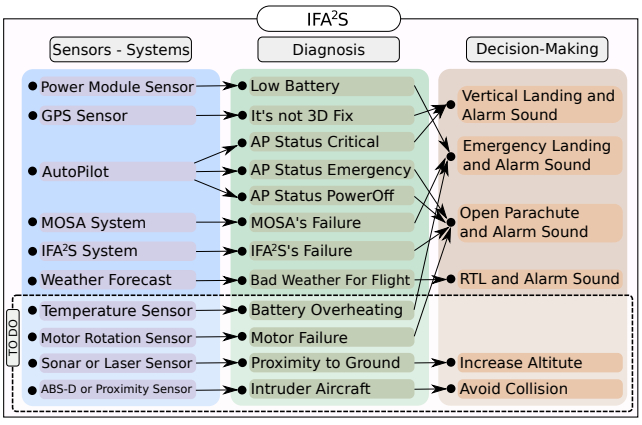

Fig. 5 Detection, diagnosis, and decision making systems of IFA²S.

50m x 36m. The simulations are executed using Software-In-The-Loop (SITL) technique.

The computer used in the experiments is an Intel i7 with 2.50 GHz, 16 GB RAM and Linux - Ubuntu 17.04 operating system. The route planning/replanning methods used were Hybrid Genetic Algorithm for the mission (HGA4m) and Multi-Population Genetic Algorithm for security (MPGA4s) developed in [6] and [5], respectively. The parameters of the route planning/replanning methods used are the same as reported in the literature [3]. The HGA4m ran within 4.0 seconds as time limit, and the assumed risk of violating an NFZ by the method was 1%. The MPGA4s performed within a time limit of 1.0 seconds, making use of up to 25 waypoints and the risk assumed by the method was 1%. The altitude of the mission was 3.0 meters.

In the first experiment, as the mission execution proposed in Section 3, it will evaluated the architecture created without considering the occurrence of a critical failure. In this study, five simulations were conducted, and their routes are presented in Figure 6. The red routes represent the routes calculated by the HGA4m method. It is interesting to note that between the points P_6 and P_7 two routes passed between the obstacles and three routes passed under the obstacles. Notice in the image that some routes are overlapping. The blue routes represent the trajectory traveled by the UAV (obtained through the GPS) trying to follow the planned route in red. In general, GPS devices have a precision error associated with each type of equipment and number of satellites captured. In this way, although this device gives a specific location, it must be questioned. In general, GPS devices have an error less than 1.5 meters radius. Figure 7 shows a flow tube representing a region of uncertainty with 1.5 meters radius at which the UAV can be located. Analyzing this figure, we realize that throughout the trajectory even with the GPS error, it does not violate the obstacles nor the NFZ.

The faults described in Figure 5 have the AP Status Critical/Emergency/PowerOff and 3D Fix diagnostics that, although implemented, were not evaluated since they are difficult to in-

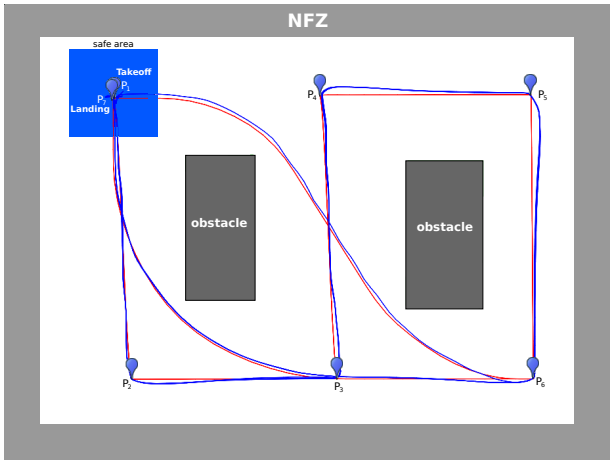

Fig. 6 Results obtained in the case study with SITL simulation.

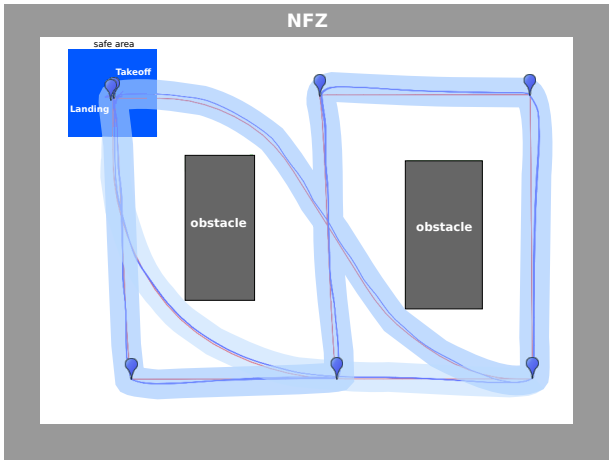

Fig. 7 Results obtained considering the routes covered with a GPS precision error flow tube.

clude in simulated environments. However, the Low Battery, MOSA's Failure, IFA^2 Failure and Bad Weather for Flight failures were fully evaluated by three times each.

The first critical fault studied is shown in Figure 8. The star represents the location where the critical fault occurred. In this way, we can see that the aircraft aborted the mission route and the UAV landed in the safe area (base) in the three evaluated cases doing the deviation of obstacles.

Another result was critical fault detection in the MOSA system. The decision made for this type of failure is to perform a replanning of routes during the flight and aborting the original mission since the system MOSA presented some internal fault. Figure 9 shows three fault simulations of the MOSA system at different locations on the route. In only one of the evaluated cases, the aircraft could not land in the safe area.

Similar to the previous experiment, some flaws in the IFA^{2}S system were analyzed. In this case, the decision-making should decide to open the parachute, once no route planning could be carried out since the IFA^{2}S presented problems. Figure 10 shows three simulations, where at F_1, F_2, and F_3 points a critical flaw occurred in the IFA^{2}S system. The landing of the aircraft using the parachute device occurred shortly after the critical failure detection by the system.

A final experiment involving failures is shown in Figure 11. It can be observed the trajectories crossed by the aircraft after the system IFA^{2}S be informed about the occurrence of bad meteorological conditions. In this way, the return to base (RTL) action was triggered. In our case, the height of the configured RTL was 8 meters, thus making the VANT overpass the obstacles found in the scenario that have 5 meters.

Some of the simulations previously described in the Figures 6 and 11 can be watched by videos highlighted in Table 1.

With the above study of the architecture, a study on the time of writing (recording) of waypoints in the AP was made. Table 2 presents the results of this study, in which it is noticed that the recording time increases linearly with the number of waypoints. This analysis reveals that, in general, it takes about 0.1 seconds to set/store a waypoint in the AP. The maximum amount of waypoints supported in AP as the pixhawk is 718.

The codes of the implemented MOSA and IFA^{2}S systems are available on the GitHub platform: *https://github.com/jesimar/UAV-Toolkit*.

6 Conclusions

This article describes and evaluated a low-cost service-oriented architecture for UAVs. This architecture was embedded on a quadcopter with an onboard computer in which several simulated experiments were performed. The MOSA software system was able to properly manage the

Table 1 Different simulations performed using SITL to validate the architecture.

Methods Evaluated	Description	Web Link
HGA4m	Full route without failure critical	https://youtu.be/kh_mH3KcHe4
HGA4m and MPGA4s (emergency landing)	Partial route with Low Battery	https://youtu.be/WQm3tn7gMs4
HGA4m and MPGA4s (emergency landing)	Partial route with MOSA's Failure	https://youtu.be/SYJopMU1Ehc
HGA4m and Open Parachute	Partial route with IFA^{2}S's Failure	https://youtu.be/-nawxwiTkiY
HGA4m and RTL	Partial route with Bad Weather	https://youtu.be/aYOpoKobXmk

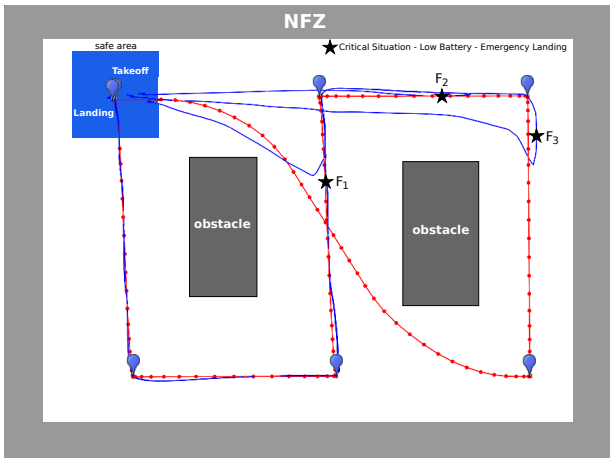

Fig. 8 Results obtained after the IFA^{2}S detect the state of low battery and then make the emergency landing.

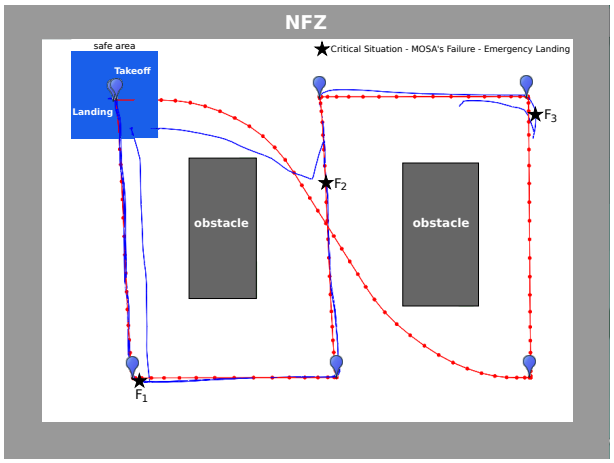

Fig. 9 Results obtained after a critical fault occurs in the MOSA system and then the IFA^{2}S makes the emergency landing.

Table 2 Summary of results evaluating waypoints writing time in AP.

Nº of Waypoints	PC Time	Seconds/Waypoint
5 waypoints	0.7 seconds	0.140
10 waypoints	1.1 seconds	0.110
20 waypoints	2.1 seconds	0.105
40 waypoints	3.8 seconds	0.095
80 waypoints	7.3 seconds	0.091
160 waypoints	14.9 seconds	0.093
320 waypoints	32.8 seconds	0.103
640 waypoints	62.9 seconds	0.098
718 waypoints	73.5 seconds	0.102

mission and route planning between waypoints safely, while avoiding obstacles. The IFA^2 system was able to react to failures that occurred internally/externally in the aircraft during the flight and managed to make the emergency landing. The quality of the routes obtained in SITL simulations was considered adequate since they avoided collisions with obstacles. The study of waypoint recording time in the AP reveals the importance of improving the (re)planners HGA4m and MGPA4s, since the updating time of in-flight routes increases linearly following the number of waypoints.

Future work intends to evaluate the service-oriented architecture proposed in real flights. It is also intended to evaluate different companions computers such as Intel Edison, Raspberry Pi and ODroid XU4 in the scenario illustrated here. It is also intended to run the IFA^{2}S system on a Real-Time Operating System (RTOS) computer as it is a critical security application.

Acknowledgment

This paper acknowledges São Paulo Research Foundation (FAPESP) for the financial support of projects 2015/23182-2 and 2014/11331-0. Research developed using computational resources of Centro de Ciências Matemáticas Aplicadas à Indústria (CeMEAI) supported by Fundação de Amparo à Pesquisa do Estado de São

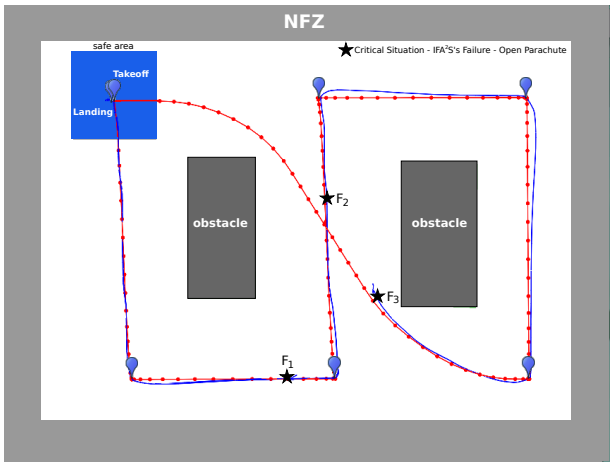

Fig. 10 Results obtained after a critical failure occurs in the IFA^{2}S system and then the parachute system is triggered.

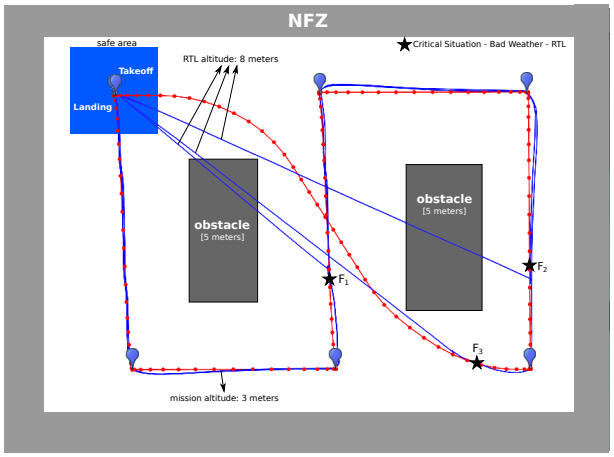

Fig. 11 Results obtained after detection of bad weather conditions on the flight and then doing the RTL.

Paulo (FAPESP) - CEPID-CeMEAI (FAPESP 2013/07375-0). This work was also supported by SENAI Innovation Institute for Embedded Systems.

References

[1] A. L. P. Mattei, A. M. d. Cunha, L. A. V. Dias, P. C. d. S. Euphrasio, O. Trindade and C. M. Toledo, *IFA2S – In-flight Awareness Augmentation Systems*. 12th International Conference on Information Technology - New Generations (ITNG), Las Vegas, 2015.

[2] N. M. Figueira, I. L. Freire, O. Trindade and E. Simões, *Mission-oriented sensor arrays and UAVs - A case study on environmental monitoring*.The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Toronto, Canada, 2015.

[3] J. d. S. Arantes, M. d. S. Arantes, C. F. M. Toledo, O. T. Júnior and B. C. Williams, *An embedded system architecture based on genetic algorithms for mission and safety planning with UAV*. Proceedings of the Genetic and Evolutionary Computation Conference, 2017.

[4] J. d. S. Arantes, V. Vaninni, M. d. S. Arantes, C. F. M. Toledo, *Low-Cost Hardware/Software Architecture for Autonomous Flight of UAVs*. International Conference on Intelligent Robots and Systems (IROS), 2018.

[5] J. d. S. Arantes, M. d. S. Arantes, C. F. M. Toledo, and B. C. Williams *A Multi-Population Genetic Algorithm for UAV Path Re-Planning under Critical Situation*. IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2015.

[6] M. d. S. Arantes, J. d. S. Arantes, C. F. M. Toledo, and B. C. Williams, *A Hybrid Multi-Population Genetic Algorithm for UAV Path Planning*. Genetic and Evolutionary Computation Conference (GECCO), 2016.

[7] M. Panahi, W. Nie and K.-J. Lin, *The Design of Middleware Support for Real-Time SOA*. 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 2011.

[8] D. Rodrigues, R. d. M. Pires, E. A. Marconato, C. Areias, J. C. Cunha, K. R. C. Branco and M. Vieira, *Service-Oriented Architectures for a Flexible and Safe Use of Unmanned Aerial Vehicles*. IEEE Intelligent Transportation Systems Magazine, vol. 9.1, pp. 97-109, 2017.

[9] J. T. Amenyo, D. Phelps, O. Oladipo, F. Sewovoe-Ekuoe, S. Jadoonanan, S. Jadoonanan, T. Tabassum, S. Gnabode, T. D Sherpa, M. Falzone, A. Hossain, and A. Kublal, *MedizDroids Project: Ultra-Low Cost, Low-Altitude, Affordable and Sustainable UAV Multicopter Drones For Mosquito Vector Control in Malaria Disease Management*. IEEE 2014 Global Humanitarian Technology Conference, 2014.

[10] A. Koubaa and B. Qureshi, *DroneTrack: Cloud-*

Based Real-Time Object Tracking Using Unmanned Aerial Vehicles Over the Internet. IEEE Access, vol. 6, pp. 13810-13824, 2018.

[11] B. H. Y. Alsalam, K. Morton, D. Campbell and F. Gonzalez, *Autonomous UAV with vision based on-board decision making for remote sensing and precision agriculture*. 2017 IEEE Aerospace Conference, Big Sky, MT, pp. 1-12, 2017.

[12] L. Apvrille, *TTool*. Telecom ParisTech, Campus SophiaTech, 2016. [Online]. Available: http://ttool.telecom-paristech.fr/. [Accessed 07 03 2018].

[13] S. Mazzini, E. Tronci, C. Paccagnini and X. Olive, *A Model-Based methodology to support the Space System Engineering (MBSSE)*. System Design & Simulation Workshop, Rome, 2010.

[14] European Space Agency, *Space project management*. ESA Requirements and Standards Division, Noordwijk, 2009.

[15] Smith J, Jones B and Brown J. *The title of the book*. 1st edition, Publisher, 2001.

[16] Smith J, Jones B and Brown J. The title of the conference paper. *Proc Conference title*, where it took place, Vol. 1, paper number, pp 1-11, 2001.

[17] Smith J, Jones B and Brown J. The title of the journal paper. *Journal Name*, Vol. 1, No. 1, pp 1-11, 2001.

Contact Author Email Address

Claudio Toledo, e-mail: *claudio@icmc.usp.br*

Copyright Statement